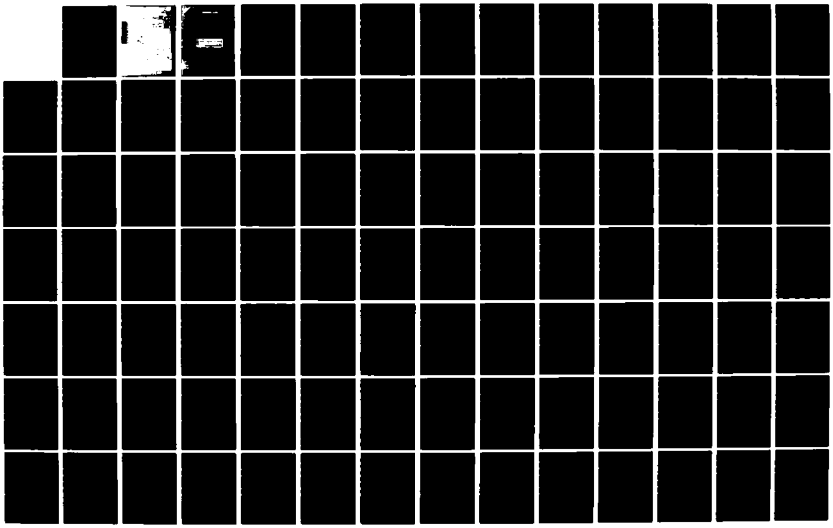
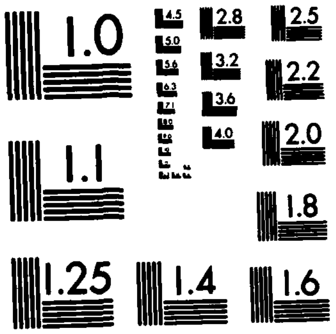
AD-A124 126 INSTRUCT: AN EXAMPLE OF THE ROLE OF ARTIFICIAL INTELLIGENCE IN VOICE-BASE..(U) BEHAVIORAL EVALUATION AND TRAINING SYSTEMS LUBBOCK TX D C CHATFIELD ET AL. JAN 83 NAVTRAEQUIPC-80-C-0061

1/2

UNCLASSIFIED F/G 5/9 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NAVTRAEQUIPCEN 80-C-0061	2. GOVT ACCESSION NO. AD-A124126	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INSTRUCT: An Example of the Role of Artificial Intelligence in Voice-Based Training Systems.		5. TYPE OF REPORT & PERIOD COVERED Final Report June 1980 - Sept. 1981
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Douglas C. Chatfield, Gary L. Klein, and David Coons		8. CONTRACT OR GRANT NUMBER(s) N61339-80-C-0061
9. PERFORMING ORGANIZATION NAME AND ADDRESS Behavioral Evaluation & Training Systems 5517 74th Street, Lubbock, TX 79424		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 3753-8P2B
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Training Equipment Center Orlando, Florida 32813		12. REPORT DATE January 1983
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DCASMA Dallas Merchandise Mart Building 500 South Ervay Street Dallas, Texas 75201		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Voice-recognition, artificial intelligence, computer-assisted instruction, instructor model, automated instruction, adaptive training, information processing, cognition, natural language, voice-based training systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Navy prototype training systems are being developed which utilize computer speech recognition technology to capture the voice behavior of trainees. These voice-based training systems have been interfaced with standard technologies for teaching and performance (continued) assessment. This current project has been targeted at examining the possibility for enhancing the pedagogical potential of these training systems through the application of Artificial Intelligence (AI) theory and technology, as well as current developments in psychological theory.

In the current effort we have sought to extend the preliminary design outlined in previous work by one of the authors, and to identify those AI technologies which would be representative of those which would facilitate the implementation of a real-world Computer Resident Automated Instructor (CRAI) in a voice-based training system. It has been noted that a gap exists between basic research and the product-oriented applications. We have attempted to help bridge that gap. To that end, we have developed implementation ideas concerning presently available theory and technology, within the constraints, as we see them, presented by voiced-based training systems.

Through the implementation of psychological cognitive theory via basic AI techniques, we have developed a simulated student whose behavior in a representative simulated task environment emulates essential features of real students. Again through the application of basic AI techniques we also developed a computer resident automated instructor which showed an ability to determine possible covert cognitive causes of overt student behavior, and to act on this knowledge to design individualized curricula.

It is felt that this investigation yields evidence supporting the logical sufficiency of the use of AI techniques for enhancing the pedagogical potential of voice-based training systems.

S/N 0102-LF-014-6601

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONTINUED)

LIST OF TABLES

SECTION I

INTRODUCTION

Navy prototype training systems are being developed that utilize computer speech recognition technology to capture the voice behavior of trainees. These voice-based training systems have been interfaced with standard technologies for teaching and performance assessment. The purpose of the current project has been to examine the possibility of enhancing the pedagogical potential of these training systems through the application of Artificial Intelligence (AI) theory and technology, as well as current developments in psychological theory.

Research in AI is dedicated to the development of technologies that give computers capabilities to behave in a fashion representative of human intelligence. The pedagogical potential of Navy instructors and training systems might be greatly enhanced if a computer-aided training system can be developed that can emulate the intelligent behavior of Navy instructors. The burden of routine training decisions could be shifted to a computer resident automated instructor (CRAI). In addition, human instructors aided by a CRAI might be able to provide a level of individualized training which manpower limitations and task complexity might prohibit human instructors alone from providing.

Chatfield, Marshall & Gidcumb (1979) reviewed recent developments in intelligent knowledge-based training systems as well as basic research in cognitive processes and the representation of knowledge. The review yielded a determination of several necessary characteristics of a training instructor and subsequently a generic model of a CRAI. Section II presents a brief reiteration of key elements of that report and reviews some more recent relevant research. The purpose of Section II is to provide a factual foundation for those readers who are new to this subject area.

Briefly, the necessary components of a CRAI, as described in Chatfield et al. (1979), included a knowledge base in which procedural as well as factual knowledge would be represented; a model of the student which would incorporate a representation of the student's evolution from novice to expert in both knowledge and processing skill; performance measurement or monitor component; a diagnostic component capable of deducing the state of covert student processing and knowledge levels from overt student behavior; and a curriculum driver that could use the student model to determine the "tutorial" function or adaptive task scenario that would optimize the student's learning. In addition, it was noted that a natural language interface would be useful in providing the student with a means for initiating information inquiries, in allowing the system to refine its diagnosis of student problems by interrogating the student, and provide the human instructor with convenient access to the system's data base on student performance and its pedagogical rationale.

In the current effort, we have sought to extend the preliminary design outlined in Chatfield et al. (1979), to identify AI technologies that would facilitate the implementation of a real world CRAI in a voice-based training system, and to provide representative examples of those technologies. Chatfield et al. (1979) noted that a gap exists between basic research and the product-oriented applications. We have attempted to help bridge that gap. To that end, we have developed implementation ideas concerning currently available theory and technology within the constraints, as we see them, presented by voiced-based training systems.

Section III briefly introduces some of the concepts in AI and relates them to some of the more traditional terminology found in psychology.

Section IV describes our symbolic generic trainer called INSTRUCT (for Intelligent Nascent Simulated Trainer for Research on Utterance Capability Trainers). INSTRUCT is an implementation of our CRAI ideas and is a testbed for those ideas. The system includes a simulated student to interact with the CRAI in the training of a task that is a simplified representation of the speech-based tasks in current Navy training systems development programs. INSTRUCT has been designed to demonstrate only the feasibility of <u>applying</u> key concepts within AI and psychology.

Westcourt, Beard, Gould, and Barr (1977) presented a detailed discussion of the use of several simulation techniques for the evaluation of computer-aided instruction systems. From their perspective, INSTRUCT is an automated simulation that uses a student simulation which is a theoretical model of student performance. With such a technique, it becomes possible to examine interactions between the CRAI and the student, which are difficult to anticipate. It also becomes possible to examine the adequacy of the CRAI's rules for inferring a student's knowledge from overt behavior. In particular, it becomes possible to compare the CRAI's model of the student with the actual covert state of the student during training because the experimenter has access to the program parameters that represent that covert state.

Westcourt et al. (1977) also highlight the limitations of evaluations with some types of student simulations. One type of student simulation presents the CRAI with a theoretically based model of the student that is an overlay of an expert model, i.e., the student is an expert with some of the knowledge or skills missing. This method can discern the CRAI's ability to infer covert knowledge states, but it can not assess the pedagogical effectiveness of the CRAI because the simulated student doesn't learn and improve its performance. Another type of simulation is to use an empirically derived statistical model of student learning and performance. This model would describe the likelihood (1) that the student learns new facts and skills as a result of the CRAI's behavior, and (2) given that it does, its behavior is such that the CRAI can discern the new learning. This technique solves the pedagogical effectiveness problem, but it implies that in order to use the simulation, the CRAI system already must have been used by real students.

We have taken another course. INSTRUCT's simulated student represents not only a theoretical model of human task performance, but also a theoretical model of human task learning. These theoretical underpinnings are based on empirical research in psychology on human learning and cognition (briefly reviewed in Section II).

In addition to eliminating the limitations of the simulation techniques above, the construction of the simulated student allowed us to examine the application of AI techniques to modeling student behavior. This experience enhanced our design of the CRAI's knowledge base, in particular its student model.

Granted, INSTRUCT's simulated student is a much simplified model of human cognition. However, our goal was not to test a full scale training system, but to examine the logical sufficiency of representative AI techniques in their application to training. To this end, the sufficiency criterion would be fulfilled if the CRAI could infer the covert cognitive states of this representative student model or display sensitivity to variations between parametrically different simulated students. Section V presents the results of our tests with INSTRUCT. We believe the results support the sufficiency of the AI techniques.

SECTION II

CHARACTERISTICS OF A VOICE BASED TRAINING ENVIRONMENT

This section reviews the analyses and theories that provided the rationale for our design of INSTRUCT. This information comes from three sources: Chatfield et al. (1979), the literature they reviewed and more recent work. Our objective is to provide the reader with a framework through which to understand the reasons behind our implementation ideas and decisions in the creation of INSTRUCT. Thus we will keep this review somewhat general. We refer the reader to the original literature for greater detail.

INSTRUCT is a simulation of a complete, albeit simplified, voice-based training system environment. This includes the simulation of a training task, a student to be trained, as well as a CRAI. Therefore, our first task was to extract the essential features of an exemplary task, the student's cognitive processes, and the student's overt behavior which we wanted to simulate.

ESSENTIAL TASK FEATURES

Chatfield et al. (1979) examined three exemplary speech-based tasks, i.e., tasks in which verbal responses are a major portion of the task behavior: Ground Controlled Approach Radar Controller, Landing Signal Officer, and Air Intercept Controller. These tasks are prime candidates for the development of voice-based automated trainers. However, there are a number of essential features which also may be found in non-speech based tasks. Training system development for such tasks may benefit from spin-offs of artificially intelligent voice-based training systems development.

First, each task is cognitively complex. These tasks require the operator to deal with a variety of informational inputs concurrently. Operators may have to attend to complex visual displays with a number of different channels of information as well as auditory input. In addition operators may be required to perform motor as well as verbal responses. The verbal response mode physically frees the operator to apply motor responses to other output channels. However, the time it takes to make a verbal response is relatively slow and improvement is limited by the physical realities of intelligibility. Therefore, there will always be some degree of concurrence between verbal responses and other events requiring attention, regardless of training. Where training will affect ability is on the cognitive requisites for making the response. We will return to this topic in our review of essential student features later in this section.

Second, each of these tasks is "event-driven." The pace of the task is controlled by external events. The operator must continually respond to changes in the task environment which are occurring in real-time. If events are not responded to quickly they may disappear or become erroneously out-of-date.

Third, the real-world event-driven rate is beyond the novice operator's capabilities. The heart of the expert operator's skills is the ability to process information from a complex display, select and execute responses seemingly in parallel. The ability to perform these tasks at a rapid pace is the major difference between the novice, and the highly skilled expert operator. Hooks, Butler, Gullen, and Peterson (1978) reports that the major determinor of an LSO's ability is the capability to perform under a seemingly cognitive overload situation without getting confused.

One feature which is specific to the speech-based task, is the current limitations on response monitoring inherent in voice-recognition systems. Robinson (1979a, 1979b) reviewed the current state of the art in voice recognition technology. The best isolated word recognition systems can correctly identify words with an accuracy of 98 to 99 percent in controlled laboratory settings for vocabularies up to 100 words. However, in the field correct identification can slip to 50 percent or less. Much of the error consists of the machine being unable to identify a word. In many environments, the machine will request a repeat speech sample rather than misidentify a word. Obviously in an event-driven task environment such interruptions would prove deleterious to the task's fidelity to the real-world and to the student's progress. From experience with such interruptions on a current trainer prototypes, one of the current authors can personally atest to the frustration such interruptions invoke.

Robinson (1979a) notes a number of sources for the error. First, there is noise. The machines tend to be sensitive both to background noise and the placement of the microphone. Another is the need for speech samples of exemplar responses to be taken by the machine. The user must repeat each word in the vocabulary often from five to ten times to the machine so that it can generate reference patterns. If, during an actual practice session, the user significantly changes the way he/she talks, the machine will botch an identification. Part of this last problem stems from the fact that people will often pronounce a word differently in isolation as opposed to during the training task. This problem becomes particularly acute when the student is under high cognitive load. As will be discussed in the next subsection, a high cognitive load will increase the probability that a student will miss-phrase an advisory or that his/her speech will become slow or degraded.

ESSENTIAL STUDENT FEATURES

The cognitive phenomena which underlie the difference in performance between novice and expert, and the phenomena which yield the evolution of the novice into an expert are of course at the core of the training effort. We turn to psychological theory for a description of these phenomena which are manifestations of the essential cognitive features of the student.

PROCESSING LIMITATIONS. The psychological literature suggests that the novice's difficulty may be traced to a limited central processing capacity. The concept of processing capacity has important implications for event-driven task training environments. It suggests that the source of student errors often may not be due to any lack of knowledge or misunderstanding of the subject matter, i.e., knowledge structure problems. Instead errors may be caused by difficulties in applying that knowledge, i.e., in cognitive processing. In our discussion of processing stages, later in this section, we will return to this distinction between process vs. structure and discuss why the pedagogical strategy applied to structure problems should be quite different than that applied to process problems.

The concept of limited processing capacity can be traced back to the attention literature of the 1950's and 1960's. The models proposed by Broadbent (1958) and Treisman (1960), for example, hypothesized that the source of the capacity limitation was in the attention stage of information processing.

More recent theories have attributed the capacity bottleneck to more general cognitive limitations. Kahneman (1973) proposed a resource competition model that assumed all cognitive tasks draw from the same pool of limited resources. Thus, if a particular task demands a large share of the resources, there are few left for other concurrent tasks. Kahneman viewed cognitive resources as a pool of "effort" which may be allocated among tasks.

Norman and Bobrow (1975) suggested that there may be two sources of processing limitation. Whenever the performance on a task can be increased with an increased allocation of resources (e.g., processing effort, or concentration), then the task is said to be <u>resource-limited.</u> Whenever the performance level remains invariant to increased allocations of processing resources, the task is said to be <u>data-limited.</u> Data-limitation can stem from two sources: external poor signal quality (e.g., low signal-to-noise ratio) called <u>Signal data-limits</u>; or an inadequately stored internal memory representation called <u>Memory data-limits.</u>

The point at which a particular task changes from resource-limited to data-limited may change with the task. The <u>performance-resource functions</u> for two tasks are illustrated in Figure 1. In this example, Task I becomes data-limited when resource allocation for that task reaches point B. Task II becomes data-limited only when resource allocation reaches just past point C. Assuming a constant resource limit, any increase in resource allocation to one task would mean a subsequent decrease in allocation to the other. Consider the situation where a student has allocated C resources to each task. If the actor increases the allocation to Task II to point D then the allocation to Task I must be subsequently decreased to point B. Given the functions in Figure 1, it can be seen that there would be no change in performance on either task. However, if we reverse the reallocation we find that performance on Task I still does not change but Task II performance declines.

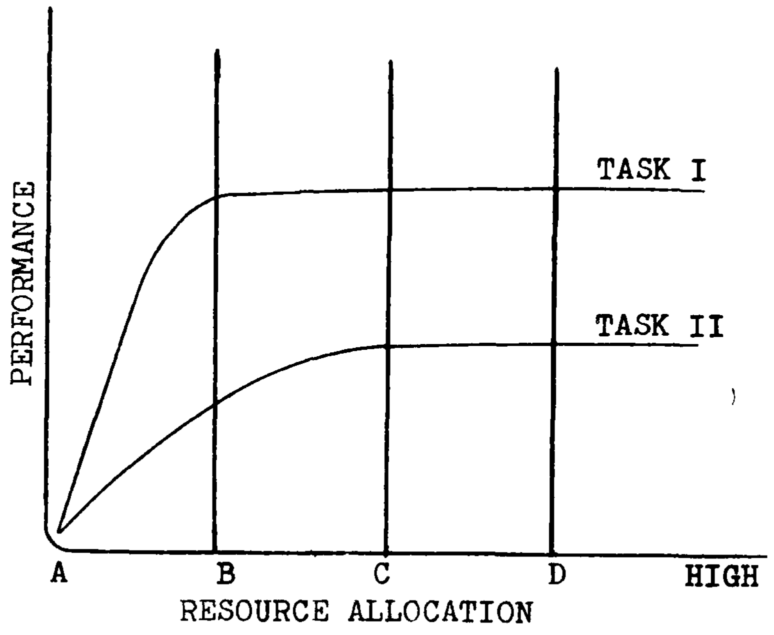


Figure 1. Two example resource allocation functions.

The change from resource to data-limitation may also be a function of task training. Norman and Bobrow explain that with practice, the students may learn to become more efficient in their processing. This increase in efficiency is graphically illustrated as a change in the performance-resource function for a given task in Figure 2. With learning the student approaches the data-limit asymptote more rapidly as resources allocations are increased. The figure illustrates that for a given performance level of a given task the expert may simply have a lot of resource capacity in reserve relative to the novice. Thus, the expert would have a greater capability to perform additional tasks. Note that even for the expert that capability is limited by the performance-resource function of each task. The greater the resource efficiency on each task the greater the number of tasks which can be handled.

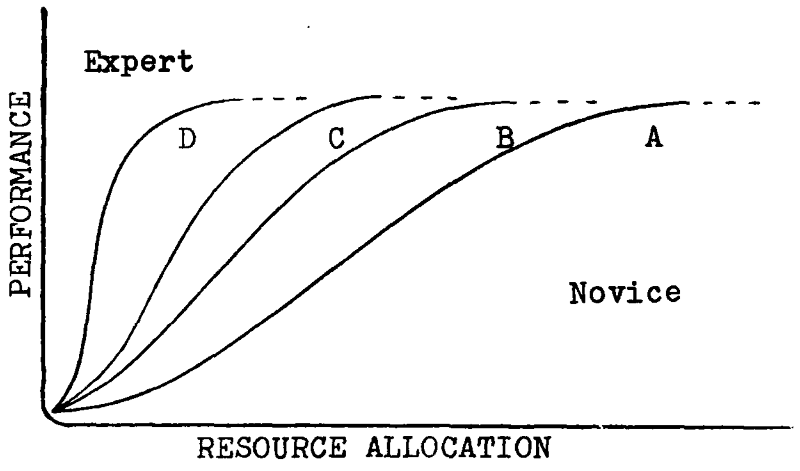


Figure 2. Changes in the resource allocation functions in various stages of training A through D.

Navon and Gopher (1979) proposed an important extension of Norman and Bobrow's theory. They proposed a theory of resource allocation based on concepts from microeconomics. According to the theory, when the composite demand of two conjoint tasks exceeds the resources available, the student must decide how to allocate the insufficient resources among the two tasks in such a way as to maximize his/her subjective utility relative to the joint performance. There may be a number of different performance mixes that will satisfy a given level of utility. This set of mixes would be described by an "indifference curve" or "equal utility contours." The importance of the indifference curves for the development of training systems is that they can be manipulated by the instructor (Chatfield et al., 1979).

Chatfield et al. (1979) noted that learning rate is also positively related to the amount of resources invested, and suggested the concept of a resource allocation-acquisition function as shown in Figure 3. The figure illustrates that the momentary learning rate is a function of the amount of resources allocated. From this perspective, the rate at which a task is mastered could be altered by manipulating the resources allocated. This manipulation could be accomplished by task priorities given by the instructor, or most directly by simply eliminating those informational inputs (e.g., blanking parts of a display screen) the instructor wishes ignored.

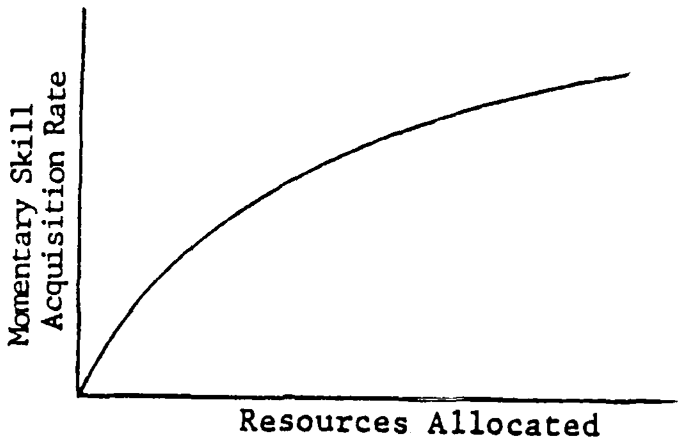


Figure 3. Exemplar resource allocation-acquisition function.

The resource allocation-acquisition function concept was empirically supported by Gopher and North (1977). Furthermore, the concept and a set of techniques for scheduling student resource allocation for optimal acquisition was suggested by Chant and Atkinson (1973) and is discussed by Chatfield and Gidcumb (1977). An important concept borrowed from economics by Chant and Atkinson (1973) is the turnpike solution. Briefly, the theory suggests that the quickest route to task acquisition may not be the most direct. For example, it may be better to train Task A in isolation, then add Task B giving it a relatively high priority, followed by a final short period of training with equal priorities, than simply training the student under equal priorities from the beginning. This control process could be extended to the decision of when to add a third task. In Section IV we will discuss our current findings resulting from a simulation study of the turnpike solution.

These considerations concerning processing limitations would suggest that an important function of a CRAI would be resource allocation scheduling. To this end, an important function of the CRAI's diagnostic routines would be to analyze overt student behavior to determine optimal scheduling. The results of our current implementation of this function are presented in Section V.

PROCESSING STAGES. Chatfield et al. (1979) reports a resurgence of interest in _stage_ theoretical approaches to the information processing problem. These approaches propose that processing takes place in discrete and independent functional stages. In addition, these stages are generally considered to be successive in nature. There is some disagreement on the nature of this succession. Sternberg (1969) proposed that only one component process may be active at any one time, a concept referred to as the _discrete stage model_. McClelland (1979) assumes that all components of a processing system operate continually, but pass information from one process to the next as it becomes available. This construction is called the _cascade model_. However, all of the stage theoretical approaches are congruent with the proposition that the reaction time of a response may be decomposed and attributed to a set of individual subprocesses (Sternberg, 1969).

Encoding or Perceptual Stage. Chatfield et al. (1979) defined _encoding_ as the process by which the raw stimulus data is transformed (or coded) into a form representative of an item in memory. LaBerge (1973) assumes a parallel hierarchical model in which the sensory input is first analyzed by feature detectors. The results of the analysis by feature detectors is then organized into codes (e.g., letters). The codes are further organized into higher level codes (e.g., words, responses etc.). Chatfield et al. (1979) suggested these hierarchies could be viewed as the _schemata_ described by Neisser (1976). Neisser (1976) defines a schema as "that portion of the entire perceptual cycle which is internal to the perceiver, modifiable by experience, and somehow specific to what is being perceived. The schema accepts information as it becomes available at sensory surfaces and is changed by that information; it directs movements and exploratory activities that make more information available, by which it is further modified." In other words, the hierarchy may be activated not only from the bottom-up via the sensory input, but also may be activated top-down via the memory system, i.e., via anticipation.

Mental Computation Stage. Once the raw stimulus data is transformed into a memory form, the next stage is to use that data to perform some task. Basic research has examined the task of search and comparison. In the classic experimental paradigm, the experimenter presents a series of characters to the subject, one at a time. The subject is to check each one and indicate whether the character is a member of a predesignated set. Work by Sternberg (1966), Schneider and Shiffrin (1977) and Briggs and Johnsen (1973) indicates that the search and comparison process is serial in nature. Schneider and Shiffrin (1977) also found that accelerating the pace of the character presentation interrupts the process and reduces accuracy. Furthermore, the effect of presentation rate interacts with the complexity of the display and predesignated comparison set.

Evidence, from both Briggs and Johnsen (1973) and Schneider and Shiffrin (1977), indicates that extensive training can reduce the effects of display and comparison set complexity. Training also decreases the effect of display pace.

Norman and Shallice (1980) have recently interpreted this effect of training in terms of their theory of "willed and automatic control of behavior." Their theory is based on the concept of sets of "active" schemata. These schemata are organized according to the particular action sequences of which they are a part. When the appropriate set of conditions occur, the schemata become selected to control action. Any given action sequence that has been well-learned is represented by an organized set of schemata. A single source schema serves as the highest order control. Other component-schemata of an action sequence can be activated via the source.

According to the theory there are three different states of a schema. The schema is normally dormant, i.e., it resides in memory but plays no role in the current active processing. A schema is activated when it is "set up, brought to a state of readiness and given an activation value." The activation value is a function of several factors: a given value from the source schema, attentional and motivation factors, the influence of other activated schemata, and the degree to which designated schema trigger requirements match the conditions of the environment. A schema is selected when its activation value exceeds its designated threshold.

From the theory's perspective, newly learned actions are apt to be ill-specified. Their schemata are relatively small, encompassing relatively specialized subactions. In addition, their triggering requirements are apt to be ill-specified: not well matched to the actual conditions that occur. The result of these deficiencies is that continual monitoring is required by attentional mechanisms, and selection must often be forced or delayed by the application of deliberate attentional activation. On the other hand, well-learned actions are apt to be well specified, with their schemata encompassing large, organized units of behavior. Their triggering requirements are likely to be well-matched to the environmental conditions. These schema can maintain control effectively for longer periods without attentional control.

Process vs. Structure. Earlier, we indicated the need for a distinction between process and structure. The state of the art is such that a rigid delineation between the two concepts is not always possible. However, in general, we will take structure to refer to the internal organization of units of a knowledge domain. As an example, in the memory search tasks of Sternberg's, the list of items in memory through which the subject must search, constitutes a simple structure. The search activity itself would be an example of what we will refer to as process.

Process. The process/structure distinction suggests that student problems may arise in either element, and that the type of instructional remedy applied should be dependent on which element is the source of the problem. Student problems which are precipitated by processing deficiencies can be attributed to resource-limitations. Therefore, an appropriate strategy would be to manipulate the student's resource allocations. Reallocating more resources to the problem process will increase performance and positively alter the performance-resource function (i.e., increase the integral of the function), by increasing the student's momentary acquistion rate.

However, if problems are due to structural deficiencies, the increases in resource allocations will not help performance, i.e., the problem can be attributed to data-limitations, specifically memory data-limitations. Reallocation under data-limitations would be wasteful by reducing resources allocated to other processes that need the enhancement and reallocating them to a process which may not. If a process is memory data-limited this factor would mask information about the state of its resource limitations. This masking highlights the fact that process and structure are not orthogonal. Just as a craftsman is limited by the quality of his tools, so the efficiency of a process is limited by the quality of its data structure. Therefore, if a student problem is attributable to data-limitations an appropriate strategy would be to determine the missing or erroneous portion of the data structure (perhaps by natural language query), then rectify the problem through factual tutorials.

We have noted that processing becomes more efficient with extensive practice. It is not clear, whether changes in process or structure underly this increased efficiency. Norman and Shallice (1980) appear to suggest that a change in the processing sequence occurs, i.e., it becomes better specified. However, efficiency might be mitigated by an increase in processing speed, or perhaps a streamlining of the data structure. For example, if a task required locating a target word in a list of words, efficiency might be increased with practice simply by increasing the speed of scanning the list. However, if the structure of the list is mutable, as is human memory, efficiency might be increased by gradually alphabetizing the list.

Structure. There has been much research effort towards discerning the nature of internal representation and memory structure. We have briefly discussed the concept of schema from a processing perspective; we have discussed the craftsman's hammer in terms of what it does, let us now describe the hammer. The term schema is used in various ways by several authors, but the common theme is that a schema is a segment of knowledge organized around a central concept (Chatfield et al., 1979). An example of a schema in its simplest form would be the frame-like notation presented by Minsky (1975) for representing knowledge in a computer. Here a <u>frame</u> is a description of an object, or action, which incorporates all the invariant features common to instances which would be classed as an example of the particular object or action.

Of present interest regarding the training of voiced-based tasks, is the relation between schemata and resource consuming events. In an event-driven situation an input generates a description which must be compared with potential schemata. Early in training this search and comparison process consumes both time and resources. If a quick match is found, the input is fit into a context which may provide further input to a higher level schema or may lead directly to a response. Thus, mismatches would be one cause for resource and time consumption. This highlights the need for developing in the student adequate schematic descriptions and prototypes during training.

Rumelhart and Norman (1976) and Norman (1978) suggest that the formation of schemata occur in three overlapping stages. During the first stage the instructional system is adding to the underlying data base of the learner. When the existing schemata cannot handle the new knowledge, the schemata themselves may be reorganized; this is the second stage. Of special interest is the third stage of "fine tuning." The schemata undergo additional minor changes, and the <u>processing</u> associated with the schemata undergoing tuning becomes streamlined, and more efficient.

Newell and Simon (1972) also indicates that the knowledge structure changes with practice. They report, for example that expert chess players can remember more chess pieces for a given board pattern in a given amount of time than can novices. However, it appears that experts process the same number of information "chunks" as do novices. The difference is that novices' chunks may include only single pieces; the experts' chunks include familiar patterns of several pieces. This merging of single informational elements into patterns of elements comes with extensive experience with those patterns.

A more recent position on the automatization of cognitive skills is given by Neves and Anderson (1981). The authors attempted to explain the mechanisms by which people accelerate in processing with practice, as opposed to simply describing the conditions under which it occurs. They propose the acceleration is due to the mechanisms of proceduralization and composition. They assert that knowledge is initially represented as a set of propositions in a semantic net, from which production rules may be built. The cognitive system, during practice creates faster production rules automatically via the procedurization mechanism. The composition mechanism forms larger units out of the individual productions. This results in an automatic acceleration, parallel search abilities, and an inability to adequately introspect on these well-learned procedures.

SUMMARY. In summary the evolution of a novice into an expert may involve qualitative and quantitative changes in cognitive processes as well as memory data-structures. We know that processing becomes more efficient with extensive training. The psychological theory reviewed indicates ways in which this change may be manifested by changes in processing speed, the manner of processing, and the data-structure in which processing takes place. Also outlined were the effects of manipulating students' resource allocation, and the need for differential training strategies regarding processing vs. structure based student problems.

ESSENTIAL FEATURES OF STUDENT BEHAVIOR ON EVENT-DRIVEN TASKS

Chatfield et al. (1979) described the typical errors of students working on a simplified Precision Approach Radar (PAR) display (only azimuth cursor was displayed). From the perspective outlined in this section (and in more detail in Chatfield et al., 1979), the vast majority of these errors would be indicative of processing limitations. For example, longer than required pauses in speech might be caused by the student not being able to speak and think concurrently. A student's slow rate of processing causes him/her to fall behind in the event-driven task which in turn causes the student to miss making some of the calls. In addition, two problems which may appear to be structural problems may in fact be precipitated by processing limitations. These are making a correct call with the wrong phraseology, and making completely incorrect calls. It was proposed that the paced nature of the task made students feel that they must respond even though they felt "hurried" and did not have enough time to fully process the response. In other words, the insufficient time (i.e., the inability to allocate sufficient resources in the time available), along with the task demands, obligated the student to produce a response which would be the product of incomplete processing. An instructor's first thought, in observing that the response was incorrect in type or phraseology, might be to conclude that the student needs remedial instruction on the rule governing the production of the response. But it was observed by Chatfield et al. (1979) that if students were asked what the correct response should have been, during a pause in the action, they could report the correct response. Hence, it would appear that the students possessed sufficient knowledge structures with which to correctly process the information, and could do so if they were not rushed or distracted. Further evidence comes from the students' comments: "There wasn't enough time," or, "Too many things were happening at once."

As mentioned in Section I, we have attempted to avoid the limitations of evaluation described by Westcourt et al. (1977). To this end, we have incorporated into INSTRUCT a simulated student which is a model of the essential student features outlined in the previous subsection. It was equally important that the simulated student emulate the characteristic errors outlined in the current subsection. Indeed, Newell and Simon (1972) suggest that the degree of fidelity with which a simulation reproduces such overt behavior can be considered evidence of the degree to which the simulation model represents the covert processes underlying the behavior. Our success in this regard is reported in Section V.

SECTION III

BACKGROUND CONCEPTS IN ARTIFICIAL INTELLIGENCE AND PSYCHOLOGY

Within the last decade, new fields and disciplines have been emerging which are bringing together researchers from diverse fields such as psychology, computer science, linguistics and philosophy. With the emergence of a new discipline comes new energy, new approaches, new societies, new journals, and, of course, new theories, concepts and terminologies. The new fields of which we speak are of course artificial intelligence (AI) and the new cognitive science. Both began quite some time ago, but in recent years have become more formalized. New journals have appeared, such as _Artificial Intelligence_, _Cognitive Psychology_, and more recently, the Cognitive Science Society's new journal _Cognitive Science_. The approaches reported in this paper emanate from the new fields and may be unfamiliar to some of our readers. In this section we will discuss some of the terminology and relate the terms to some of the more traditional terms found in psychology. The terms will be only briefly introduced, however, and the reader will have to consult some of the newer textbooks for more comprehensive discussions. Throughout the paper, numerous sources will be cited which would be good references.

Artificial intelligence has been described as "the study of ideas which enable computers to do the things that make people seem intelligent" (Winston, 1977, pg. 1). It might then be asked, "What is it that makes humans intelligent?" Moreover, "What is intelligence or thought?" In the past, traditional psychology, laboring under its heritage of S-R associationism, had focused on a fairly narrow set of definitions of learning, memory perception and problem solving. In the past decade or two, these traditions have been broken, leading to a new revitalized rise in cognitive psychology which was once seen as having been stamped out by behaviorism. With contributions from the cognitive psychology movement, AI, linguistics, and philosophy, the new cognitive science has emerged. Simon (1981) believes that the trend started in 1956 with Miller's (1956) information processing, Chomsky's (1956) transformational grammars and Bruner, Goodnow, and Austin's (1956) work on thinking. This new science of cognition has been called a science of the mind, of intelligence, of thought; a science concerned with knowledge and its uses. To put it in the words of Norman (1981):

> Cognitive Science is the search for the understanding of cognition, be it real or abstract, human or machine. The goal is to understand the principles of intelligent, cognitive behavior. The hope is that this will lead to better understanding of the human mind, of teaching and learning, of mental abilities, and of the development of intelligent devices that can augment human capabilities in important and constructive ways. (pg. 1)

The division between cognitive science, AI, and cognitive psychology is quite blurred as they seem to be blending and working toward common goals, differing only in emphasis. Although our intent is to relate the terms in cognitive psychology to those in AI, at times it is hard to tell where one area ends and the other begins.

The scope of AI is somewhat hard to define. The topical categories used at the first annual conference of the American Association for Artificial Intelligence (AAAI) held at Stanford University in August 1980, were essentially the following: Vision, Theorem Proving, Problem Solving, Knowledge Representation, Knowledge Acquisition, and Natural Language. In a sense, these categories might be said to represent the field, although some of the textbooks have organized the field a little differently. Of the topics listed, we will be concerned primarily with the last four in our conceptualization of the CRAI.

KNOWLEDGE

The representation and acquisition of knowledge is probably the most basic to all areas of cognition and AI. Historically, psychology always has been interested in the way people acquire knowledge, but it was the philosopher that was interested in the nature of knowledge. Experimental psychologists were interested in people as processors of information: the way they acquired knowledge from information, the way they stored it and they way they retrieved it. They hypothesized the existence of several repositories of information, such as sensory memory, short-term memory and long-term memory. In 1972, Endel Tulving published Organization of Memory, proposing a distinction between episodic and semantic memory as researchers were beginning to take a great interest in the structural contents of memory and its relation to retrieval. Reaction time was used as a dependent variable to reflect the manner in which the material was organized. Collins and Quillian (1969) put forth a network model of semantic memory as did Anderson and Bower (1973), Kintsch (1974), and the Active Structural Network model of Norman and Rumelhart (1975). The network models, such as the Collins and Quillian model, assume that information is hierarchically organized such that each node, representing a concept, is related to other nodes in a subordinate-superordinate fashion as shown in Figure 4. With each node is a set of properties which would characterize that node and all of its subordinates. Important to our conceptualizations in automated training was the idea that the traversing of the network consumed time. Support for their work came from experimental evidence in the form of reaction times. Subjects were asked to verify propositions such as: "A canary eats. True or false?"

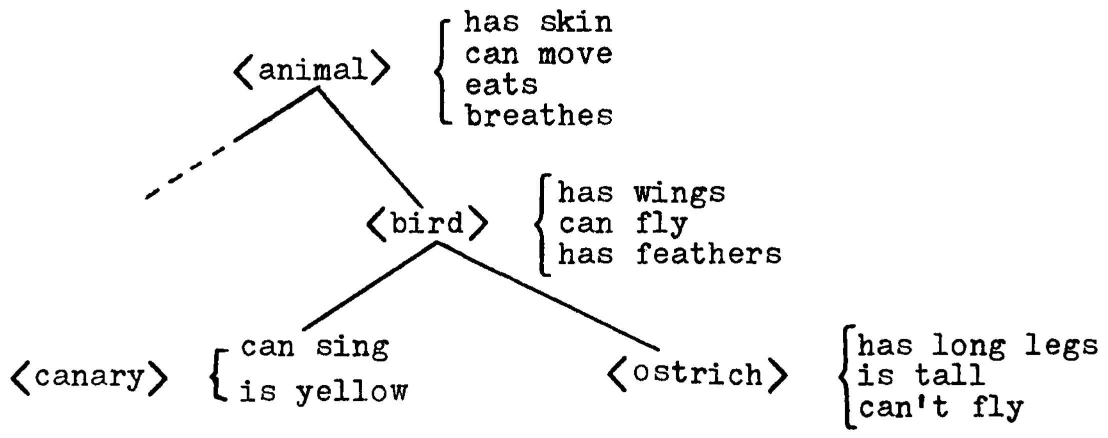


Figure 4. A portion of a semantic net from Collins and Quillian (1969).

During this period, the semantic net was being utilized as a way of representing knowledge in an intelligent computer assisted instruction (ICAI) application by Carbonell (1970). The training system, called SCHOLAR, represented declarative facts about geography as a semantic network and offered a different form of validity for the idea of a network model. Other network-based ICAI systems are reviewed in Chatfield, et al. (1979).

Alternatives to the network models were the set-theoretic models and the feature-comparison model. An example of the first type was that of Meyer (1970) in which knowledge consisted of a set of defining attributes. An example of the latter model is given by Smith, Shoben and Rips (1974) in which membership in a category is determined by a set of defining features or attributes similar to the set-theoretic models.

AI researchers during this period, also were interested in organizing information for computing systems and were using similar approaches. Minsky (1975) proposed his theory of frames, bearing the influences of the semantic network models.

To Minsky, a frame was a data-structure used to store information that was typical of the situation or concept which the frame was to represent. The data structure contained terminals or slots which were to be filled with the characteristics of the situation. Some of the slots already were filled with things that were always true, while others were empty and were filled by data when a particular instance was being referenced (instantiation). The slots themselves also contained a set of conditions required for assignment of entries. One of the more powerful features of frames was that the slots could be filled with default assignments (things assumed to be true) which could be easily replaced if additional data was forthcoming. Individual frames were linked together into frame-systems, which in turn were linked by an information-retrieval network. Frames, then, are a convenient way of organizing information about the world (domain) in which the intelligent system is to function. Other knowledge structures referred to are scripts, goals, and plans (see Schank, & Abelson; 1975) and affective units.

NATURAL LANGUAGE

One cannot functionally separate language from knowledge in an intelligent system, yet we have become accustomed to dividing broad fields into pedagogically convenient groupings. Such a division is often made between syntax and semantics. Syntax refers to the comparatively local and superficial aspects of sentence structure whereas semantics refers to more global and deeper concerns. Syntax involves the rules for the sequencing of word units and, together with morphology, constitute the more formal grammars. The grammatical theories with which we would be most familiar would be the phrase structure grammars and, of course, Chomsky's transformational grammar. The transformations refer to the changes that can be made on an underlying string or sentence referred to as the kernel string. It is in the syntactic analysis of strings that we encounter parsers and augmented transition nets which we will discuss in Section IV in relation to INSTRUCT's interrogatory interface.

Language understanding comes with the study of semantics. Munro (1975) refers to two basic approaches to semantics: the structural approach of the LNR Research Group and the approaches taken by generative semanticists. This latter group is mainly responsible for the decomposition of semantics into a set of primitives. Lakoff (1970) is cited as the one who introduced the use of "primitive" predicates. Other work in lexical decomposition, and perhaps more widely used now, is the conceptual dependency analysis of Schank, in which he proposed a different set and number of primitives. Reduction to a set of primitives is beneficial in reducing the size of storage of the knowledge-base, since everything could ostensibly be understood in terms of these few primitive acts and in the handling of paraphrases. The LNR Research Group, in contrast, has taken a propositional approach. Here the parts of speech, other than verbs, are represented as predicates.

PROBLEM SOLVING

Traditionally psychologists have used both the associative and Gestalt approaches to problem solving, but neither have been sufficiently comprehensive to serve as a framework by which to understand problem solving. Most of the progress has come through comparing human processing to that of the computer (see Hunt, 1971; and Newell & Simon, 1972). Problem solving is a complex and many-faceted problem. Researchers in their limited scope have chosen to highlight only limited facets at any one time. Of interest to us are those areas relating to problem solving in AI and the use of the computer as an expert system.

Human problem solving involves the encoding of information, and the modification or transformation of that information, and possibly some sort of necessary action. The transformation of the information and subsequent operations is likely to involve the retrieval of factual and procedural information from long-term memory (LTM) as well as the use of short-term memory (STM) as a working buffer. Greeno (1973) divides the contents of LTM into two parts. He makes a distinction between propositional knowledge and algorithmic knowledge, with both categories being a part of semantic memory.

Another approach to the study of problem solving is to look at the set of algorithms or approaches humans bring to bear. Psychologists were interested in the area because of their interest in the "typical" non-optimizing or heuristic approach used by people whereas those working with computers were interested in optimizing the process to solve what the human could not. It became apparent however that the heuristic approach used by people, though inefficient or suboptimal on a specific problem, was applicable to a great range of problems. Additionally, the AI community was interested in emulating human behavior as an end in itself.

The classic work in this area is Newell and Simon's General Problem Solver (GPS). With this work and others came a series of concepts and techniques that have since been widely used. GPS had a particular strategy for solving problems: the means-end-analysis. Through the analysis of the difference between a current state and the goal state, GPS used a control regime by which operators were selected which would reduce the observed difference. GPS broke the ground with some concepts and techniques that have enjoyed wide implementation since then. One implementation of GPS is the STRIPS system developed at the Stanford Research Institute. Other problem solving systems have been used in various applications such as DENDRAL, and MYCIN.

A number of principles have evolved in AI, which cut across various applications and have their roots more in computer science and mathematics than in psychology. For that reason, we mention only a few of the more important principles in passing. Most of the AI systems are organized into three standard components: a global database, a set of operations and a control system. Such is the case for a computational formalism known as a production system. Here the operations are a set of production rules directed by a control system. Briefly, the production rules are if-then rules which operate on the data base. The control regime is a set of strategies by which certain production rules are applied, such as difference reduction in means-end-analysis; but in studying the path taken by the control regime's selection of a sequence of production rules as it traverses a network or tree structure, we are looking at search methods. The various common search methods, such as, depth first, breadth first, best first, hill climbing, branch and bound, and the pruning techniques, are discussed in Winston (1977) although a more thorough discussion can be found in Nilsson (1980).

LISP

Most of the developments in AI have taken place in environments with considerable computing power, and with specialized languages. Although, as Fahlman (1981) points out, good AI research has been on machines with hostile (to AI) operating systems and in languages ranging from BASIC to PL-1. As always, the most productive work gets done in the most conducive and supportive environments. The languages in use in AI development are all symbol manipulation languages, such as SAIL, POP-2, LISP, and, of course, IPL, which preceded them all, developed by Newell, Shaw and Simon. LISP is the most widely used and serves as a foundation for others, such as PLANNER, CONNIVER, and PLASMA.

Because developments in AI are so intrinsically involved with LISP, an overview of some of the terms would not be complete without a brief discussion of the language in which most of the AI concepts are implemented. Because of our brevity, however, there are several texts and manuals which the reader may want to consult. We would recommend Allen (1978), Siklossy (1976), Weissman (1967), and Winston and Horn (1981).

LISP is a symbol manipulation language derived from a part of mathematical logic known as recursive function theory. The acronym refers to "LISt Processing." There may be some disagreement as to why LISP is so widely used, but there seems to be little disagreement about its dominance over the other languages. In fact, at the first AAAI conference at Stanford, LISP seemed to have an overwhelming number of advocates widely proclaiming its virtues (like disciples). Winston and Horn (1981) cite four possible reasons for its wide acceptance:

1. It has the convenience of being interactive.

2. It has the best in editing and debugging tools for writing large, intelligent programs.

3. It possesses an optimal set of features for symbol manipulation.

4. LISP functions can be used and treated as data. In fact, one function can assemble another and use it.

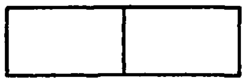
It might be pointed out that the advantage of LISP is primarily in the experimental development of AI techniques and theory and not necessarily in application. Intelligent programs, once developed in LISP, can be translated into other languages for particular applications. Thus, LISP grew up in academically-oriented environments with their considerable computing power and ability to create LISP to conform to their needs. Fahlman (1981) gives a good survey of the computing facilities that would be required for AI, although it must be kept in mind that he speaks mostly for AI research centers. Commercial users of certain AI techniques would undoubtedly have different requirements. In reviewing the computer needs, Fahlman discusses the time-sharing environments separately from the personal-computing options. If working within a time-sharing environment, the Digital Equipment Corporation System-20 family with a TOPS-20 operating system or the VAX family are recommended; but the personal-computing option seems to be the recommendation of the future. This is the use of specialized LISP machines, dedicated to a single user, though they may be connected to each other via a network. These are the machines that were developed and used by some of the major research centers and should become available for purchase during 1981. Their sale price seems to vary between $30K to $150K. Fahlman lists various advantages for the dedicated machines as well as listings of the present and future suppliers.

LISP is a language where statements and data are both treated as symbolic expressions called S-expressions. These S-expressions are made of lists of elements or even lists of lists. The most basic element is referred to as an atom, which may be a single character, a character string of indefinite length, or even a number. For example, both of the following are S-expressions:

(NOUN AGENT SING)
(VERB (TENSE (PRES)))

but they are represented differently internally. Technically, an S-expression is either an atom, e.g. NOUN; a pair of atoms, e.g., (NOUN-AGENT); or a pair of S-expressions, e.g. (NOUN. AGENT. (SING))). The parentheses and dots are a way of representing the S-expression in what is called dot notation. The dot notation illustrates that all non-atomic S-expressions are actually represented internally as a binary tree structure.

Another means of representation of S-expressions is graphically. To illustrate, let the following symbol

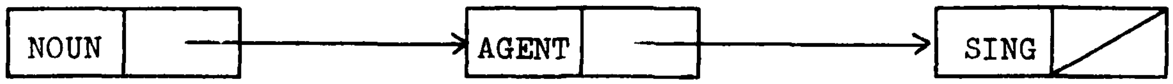


represent a graphical node with a left and right branch. Further, we will let arrows represent pointers to another node or atom, i.e., a unique machine address.

With this notation, the list

(NOUN AGENT SING)

would be represented as:

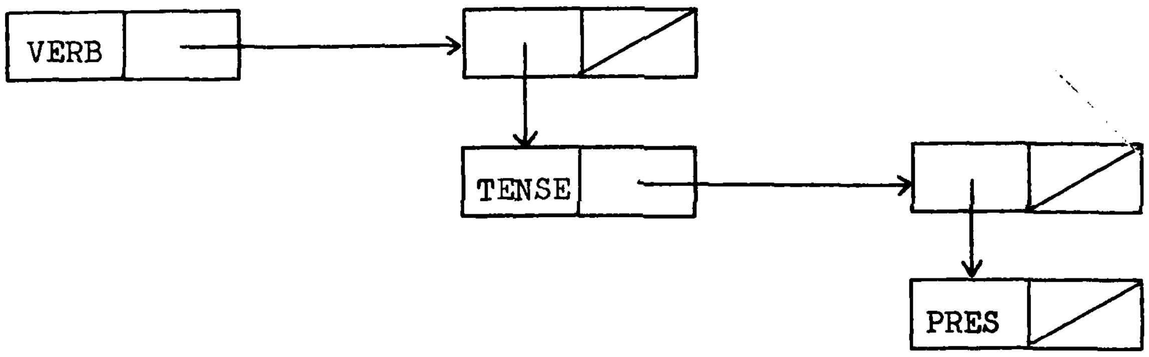


where the diagonal line in the last box represents NIL which is both the empty list and an atom. We introduce this symbolism because it is used in some of our citations. In fact, it appears in the logo of the LISP Machine Corporation, one of the suppliers of a personal LISP machine.

It is worth noting that the expression:

(VERB(TENSE(PRES)))

represents a list of (an atom paired with a list (wherein the list in turn consists of an atom paired with a list (which in turn consists of an atom))). The graphical representation of this expression would be:



The structure of the graph is meant to illustrate the way lists are internally represented as binary tree structures.

LISP has numerous "built-in" functions which take lists apart, put them together, and generally manipulate them. A few selected examples of such functions will be discussed for illustration. To begin, one could name a list by using the function SET:

(SET 'VERBS' SEE LOOK PERCEIVE)

which would associate the list (SEE LOOK PERCEIVE) with the atom VERBS.

Furthermore, we could seek the first element of a list as shown:

```
(CAR'((SEE LOOK PERCEIVE))
```

returns SEE,

```
(CAR'((SING PLURAL) NUMBER))
```

returns (SING PLURAL), or we could delete the first element, keeping the rest as in

```
(CDR'(SEE LOOK PERCEIVE))
```

returns (LOOK PERCEIVE).

We also could seek the middle element by putting together a combination which would take the first element of the list, that results from the use of CDR as above.

Thus, in essence, CADR the CAR of a CDR as in:

```
CDR'(SEE LOOK PERCEIVE))
```

returns (LOOK PERCEIVE),

```
(CAR'(LOOK PERCEIVE))
```

returns LOOK. Therefore,

```
(CADR'(SEE LOOK PERCEIVE))
```

returns LOOK.

In addition to these functions, others can be used to construct lists, such as APPEND, LIST, CONS, and DELETE.

A different class of functions are the predicates whose values always are T (true) or NIL (false). Such is the case for MEMBER, which returns T only if the first argument in the function is a member of the second argument, which always is a list.

```
(MEMBER 'THIS '(THIS IS A PLANE))
```

returns T.

```
(MEMBER 'THE '(THIS IS A PLANE))
```

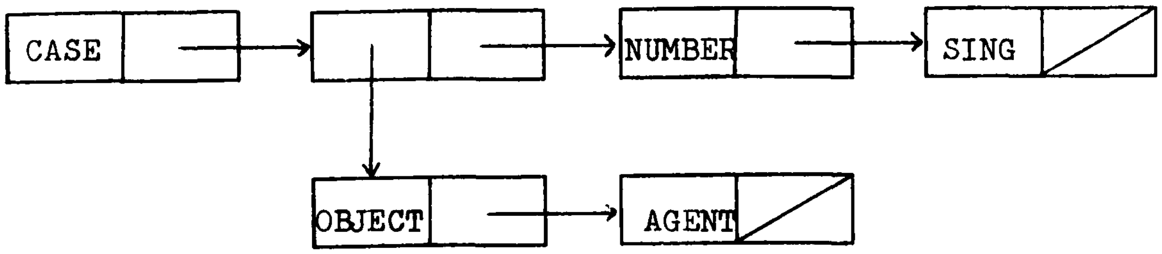
returns NIL.

Numerous other predicates exist which evaluate whether two expressions are equal, whether an S-expression is an atom, whether a number is even or odd, or whether it is less than or greater than another number, etc.

The power of LISP begins to be appreciated when one begins to look at the property list functions. Take, for example, the noun "pilot." Its property list might be listed as the following:

```
(CASE (OBJECT AGENT) NUMBER SING)
```

wherein the graph structure would be:



One of the property list functions is GET, which searches the property list of the first argument for the entry listed as the second argument. If it is found, then its property -- the next list element (the CAR of the rest of the list) -- is eturned as the value of GET.

Thus:

```
(GET PILOT 'NUMBER)
```

returns SING.

If you want to know if the "pilot" could possibly be a noun of the agent case,

```
(MEMBER 'AGENT (GET PILOT 'CASE))
```

returns T. This is because the GET function returned the next element that followed CASE, which was the list (OBJECT AGENT). The MEMBER predicate simply returned T because AGENT was in fact a member of that list. Other property list functions include a PUT function, which allows properties to be added, or the REMPROP function, which removes properties.

Another feature of LISP is that it will allow the defining of functions, a feature which is widely used in most intelligent programs. In fact, it is used so much that a section of code may look like a completely different language. Its flexibility allows functions to be used as arguments and as data, i.e., elements in a list. This means that it has an ability to assemble the elements of a new function on its own. Additionally, LISP possesses the capability for considerable recursion, i.e., an ability for functions to call themselves. As an example, we will illustrate both recursion and the defining of functions by demonstrating the defining of the predicate MEMBER which was described earlier.

Before we begin, we need to explain two more predicates and an additional function. The first predicate is ATOM, which takes only one argument. ATOM returns T if the argument is an atom. It returns NIL if the argument is a list.

The second predicate is EQUAL which takes two arguments. As one may expect, EQUAL returns T if the two arguments are identical S-expressions. It returns NIL if they are not.

LISP also has a conditional statement of indefinite length with the following syntactic form:

$$(\mathrm{COND}\ (p_1\ e_1)\ (p_2\ e_2)\ -\ -\ -\ (p_n\ e_n))$$

where p_i refers to a predicate and e_i to a functional expression. When evaluated, if p_1 is not NIL, then e_1 is evaluated and the process leaves the COND function. But if p_1 is NIL, then e_1 is not evaluated and the process goes on to p_2. If p_2 is not NIL, then e_2 is evaluated, otherwise, on to p_3, etc.

Now we can define MEMBER in terms of ATOM, EQUAL and COND. We will use the following logic, which is presented first in pseudo code:

```
Define MEMBER(word, list)
   If list is already an atom,
     Then: If atom is equal to word,
             Then: Return T.
             Else: Return NIL.
     Else: If word is equal to the
           first element of list,
             Then: Return T.
             Else: If word is MEMBER of
                   the rest of the list
                     Then: Return T.
```

It is a simple definition except for the last "If" statement, in which the function is used in its own definition. In LISP notation, it would read:

```
DEFINE (MEMBER WORD LIST)
       (COND ((ATOM LIST) (EQUAL WORD LIST))
             ((EQUAL WORD (CAR LIST)) T)
             (T (MEMBER WORD (CDR LIST))))).
```

Notice that in the last line the function that we are defining is in fact being used in that definition. Recursive examples like this can be demonstrated in other languages like BASIC, FORTRAN, etc., but are much more difficult to implement.

There are many other features and concepts that are notable in LISP. The reader is encouraged to consult the sources cited for explanations of other functions, predicates and special forms. Not discussed were such features as Lambda functions and the binding of variables in a local region (scope) of the program. Also not discussed were any editing or debugging tools, input-output features, or the capabilities for storage and garbage collection (a colorful term to describe the reclamation process in internal storage management). Our intent was not to create a small LISP manual when others exist, but only to introduce some of its features within the context of intelligent instructor models for voice-based trainers. It is hoped that LISP features and the other concepts from AI will serve as an introduction to the approach and concepts we advocate for an intelligent instructor model.

SECTION IV

INSTRUCT

OVERVIEW

INSTRUCT is a symbolic generic training test system. We refer to it as a test system because it includes a training task, a trainee and a CRAI. The term symbolic refers to the fact that the training task and the trainee are both simulated. There are no actual visual displays watched, no words actually spoken, and the events occur in simulated time. The term generic refers to the fact that the task environment is meant to represent the essential features of an amalgam of event-driven training task environments. We wished to maintain the generality of the techniques, algorithms and findings developed in this effort. Figure 5 illustrates the general flow of the system.

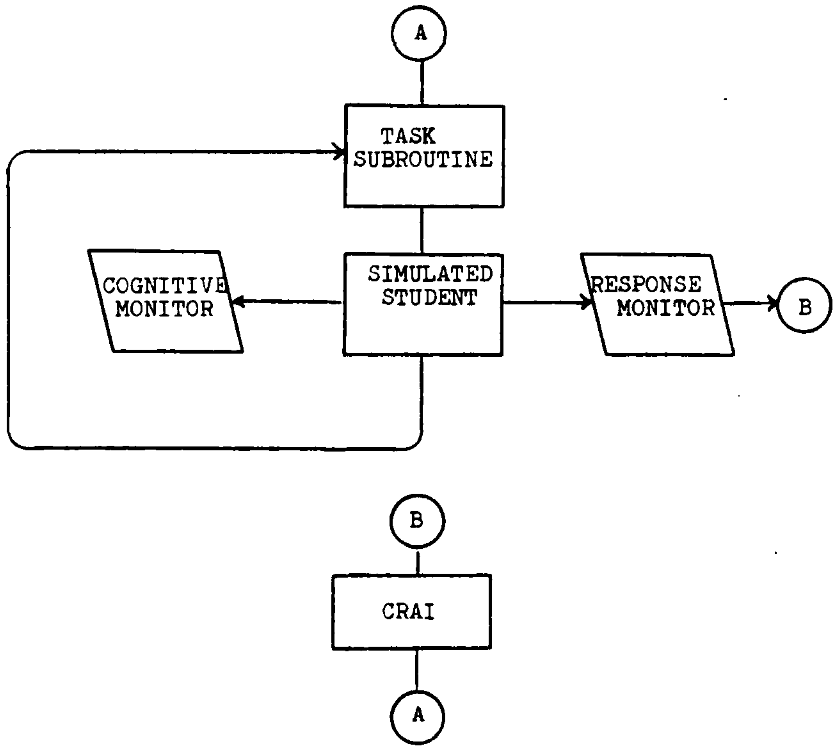


Figure 5. General flow of INSTRUCT.

The CRAI pedagogically determines the parameters for the practice task and passes these to the task-student loop. Using these parameters the task subroutine calculates a display value and makes this parametric value available to the simulated student subroutine.

The simulated student subroutine completes one processing act in light of its internal states. It then outputs a subsequent set of parametric values to the monitors. One of these monitors records the student's "covert" cognitive parameters. These will be used by the experimenter for analysis of the CRAI performance. Student's cognitive parameters are recorded every time the simulated student completes some cognitive processing act. The other monitor records only the student's "overt" responses, representative of the type of information available to a real-world voice-based training system. This information will be used by the CRAI for its analysis of the student and subsequent pedagogical decisions. Overt responses are recorded, obviously, only when a processing act results in an overt output. The time parameter is advanced at the completion of each processing act. This incrementation is an act-specific mathematical function of a number of student factors, e.g., learning level and resources allocated to the act. This time parameter is passed to the task subroutine which updates the display value accordingly. The task-student loop continues iterating until the practice task is completed.

Once the practice task is completed the CRAI then analyzes the student's overt responses, makes a pedagogical decision as to the task parameters for the next practice task and passes these to the task subroutine. A complete training program consists of five practice tasks. The CRAI itself is not a simulation. The inputs it receives and the outputs it makes are of the same quality, albeit not the same complexity, of a real-world CRAI. Similarly, the processing steps it takes and algorithms it uses are representative of those applicable to the real-world. It is, after all, the logical sufficiency of these steps and algorithms we are interested in examining.

Although the system is symbolic, we have found exposition is aided by discussing various phenomena and components as if they were real. Thus we find ourselves discussing the student "scanning the plane" or "not seeing it pass a call point." Since such "concrete" descriptions truly facilitate visualizing what the system is doing we shall adopt this mode often in the following descriptions of INSTRUCT components.

THE TASK

The task we designed is meant to be representative of air traffic controller type tasks. As such, it incorporates the essential task features outlined in Section II.

The following is a concrete description of the primary task. Our student sits watching a display tube. The display is a PAR type elevation glideslope display (see Figure 6). The cursor represents the designated glidepath. In our task each hashmark represents a mile mark, and the student must make an elevation call as the plane passes each mark. An approach consists of the movement of the target along the cursor from the right end to the left end. Thus the student is supposed to make ten calls per approach. The call the student must make is dependent upon the point at which the target intersects the cursor. For illustration, the target is broken into seven zones in Figure 6.

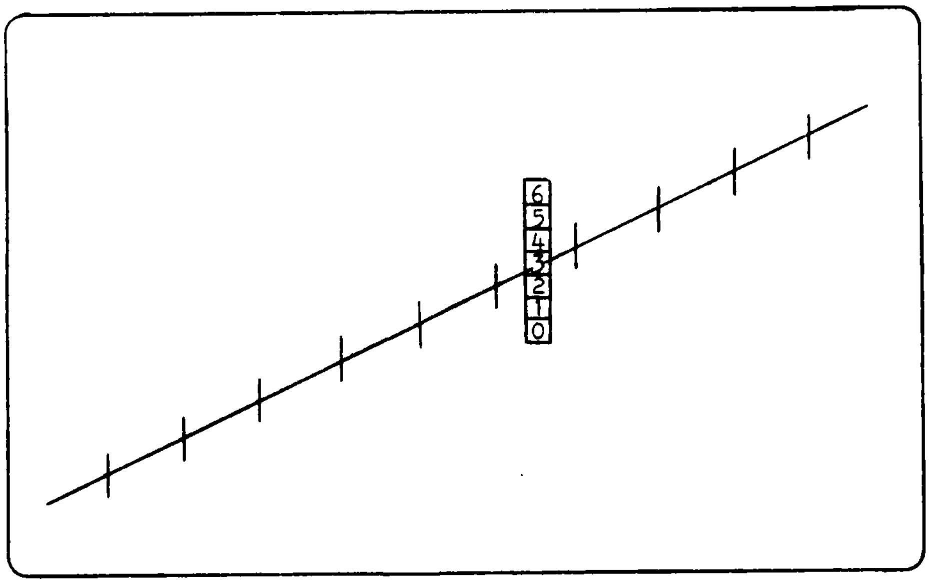


Figure 6. INSTRUCT task display.

The call corresponding to each zone is given in Table 1. these zones are not actually demarcated for the student. A significant part of the training task is to train the student to discern the correct zone.

TABLE 1. GLIDEPATH ELEVATION ADVISORIES.

ELEVATION ZONE	ADVISORY
6	Well Above Glidepath
5	Above Glidepath
4	Slightly Above Glidepath
3	On Glidepath
2	Slightly Below Glidpath
1	Below Glidepath
0	Well Below Glidepath

There are five variables which define an approach <u>scenario</u>. First, is the average glidepath elevation. The plane can be set to vary around a particular glidepath zone. Second, is the starting glideslope trend of the the plane when it first appears on the screen. The plane can be set to begin by rising or falling. Third, is the maximum degree to which the plane will deviate from the average glidepath elevation. Fourth, is the frequency with which the plane crosses the average glidepath elevation. One might consider this a pilot ability factor: less experienced pilots tending to overcorrect. Fifth is the speed of the plane. Because the task occurs in simulated time, the speed of the plane is relative. Any attempt to translate it into real time would be arbitrary. We will explain how these factors interrelate and how they are manipulated to achieve the optimum scenario in our discussion of the CRAI components.

Although it is considerably simpler than full-scale real-world tasks, INSTRUCT's task does contain the essential features outlined in Section II. First, it is cognitively complex. The student must attend to vertical position, discern the horizontal position, retrieve the appropriate verbal response, and produce that response concurrently. Second, the task is event-driven. The pace of the task is controlled by the speed of the plane. If the student fails to respond quickly enough the plane may move from the elevation zone the student is trying to process, or the plane may pass a designated call point before the student is fully prepared to make a call. Third, the event-driven rate can be made to be beyond the novice's capabilities while still within the capabilities of a simulated expert. This last fact is revealed by our simulation results which will be discussed in greater detail in Section V. With training, our simulated student can achieve a perfect performance on an approach scenario which is totally beyond the students initial capabilities.

THE SIMULATED STUDENT

OVERVIEW. Our simulated student is a hierarchical frame-system. Figure 7 presents a general illustration of this system. The system is conceptually similar to that currently proposed by Norman and Shallice (1980) and Norman (1981) but was developed independently. For the sake of clarity and conceptual parsimony, we will defer at times to their terminology.

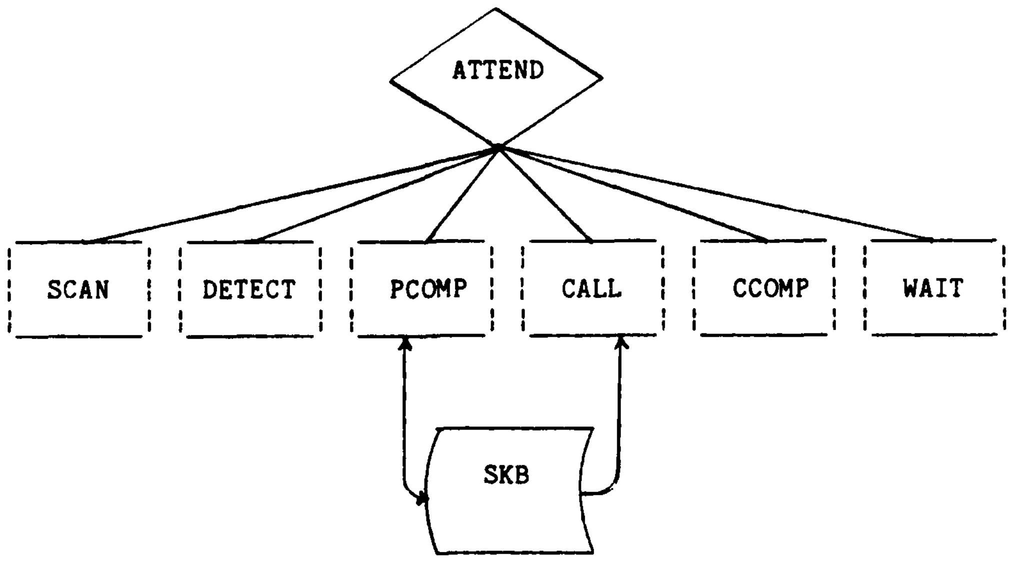


Figure 7. The simulated student.

There are six process frames we will call PROCS. Briefly, ATTEND is the central control frame: it tests the student's states and environmental information available to it and determines the next cognitive action to be taken. SCAN is the student's feature detector schema: it makes environmental information available to the student. DETECT is a primary level encoding schema: it spatially discerns plane elevation levels. It almost always follows a SCAN. PCOMP is a mental computation schema: it uses the student's knowledge base to encode the spatial information from DETECT into a higher level verbal response. CALL is an action schema: it directs the student's overt responding. CCOMP is a feedback schema: it directs the student's analysis of the overt responses and determines possible changes in the student's cognitive state. WAIT is a default schema: WAITing is what the student does when encoding of the verbal response is completed, but it is not time to make an advisory.

As alluded to above, the student also possesses a data-structure which we will call the student knowledge base (SKB). It is also, initially, a hierarchical frame-system. This data structure in its initial construction is conceptually outlined in Table 2. We qualify this illustration as initial because this data-structure evolves into a homogeneous single frame as training progresses.

ATTEND. This frame directs the student's consideration of the next cognitive act to perform. Its decision is a function of a number of variables. These include: the last act performed, the vertical position of the plane (i.e., its relationship to the call points), and chance. For example, ATTEND will not select CALL if a CALL has too recently been made (i.e., the student will not make two advisories for the same call point). Given this proscription the probability of selecting a CALL is a function of the closeness of the plane to the call point. This probability increases to an asymptote of 100 percent as the plane passes the call point. However, if the student is busy performing some cognitive act while the plane passes the call point the student might not readily ATTEND to the plane's vertical position in time, causing the student to either make the advisory late or miss it entirely. The student's consideration of some acts may override its consideration of others. For example, the student normally will DETECT immediately following a SCAN. However, if the SCAN occurs near a call point, the student may interrupt this sequence and perform a CALL instead. In

TABLE 2. HIERARCHICAL STRUCTURE OF SKB.

MAIN: Lowest Resolution Frame.

Position	Next Frame	Learning
Somewhere Above	ABOVE	0
Somewhere Above	ABOVE	0
Somewhere Above	ABOVE	0
On Glidepath	ACTUAL	0
Somewhere Below	BELOW	0
Somewhere Below	BELOW	0
Somewhere Below	BELOW	0

ABOVE: 2nd Resolution Frame.*

Well Above	ACTUAL	0
Above/Slight	AS	0
Above/Slight	AS	0

AS: 3rd Resolution Frame.**

Above	ACTUAL	0
Slightly Above	ACTUAL	0

*BELOW is mirror reflection with positions and frame pointers below the glidepath.

**BS is mirror reflection for below the glidepath.

Norman and Shallice's (1980) terminology ATTEND is the source schema which activates the other PROCS causing their selection.

SCAN. This frame directs the student's sensing of the environment. It is the process through which raw information about the task display enters the student system. SCAN's selection by ATTEND involves chance. All things being equal, the closer the plane is to a call point the greater the probability of SCAN being selected. Once selected, this frame updates the student's information concerning the vertical and horizontal position of the plane. It will also cause an emergency flag to be raised if this information indicates that an advisory has been missed. SCAN is a sensory act. SCAN's output is isomorphic with the environmental information. Therefore, there is no error, and subsequently no learning in this frame. However, the time it takes the student to do a SCAN decreases with training. This decrease is a function of the resources allocated to SCAN and the number of times the frame is selected.

DETECT. This frame directs the student's primary spatial encoding of the plane's glidepath elevation. It will almost always be selected following a SCAN. The exception is when a CALL is given priority. DETECT encodes the raw data made available by SCAN. DETECT can make an error in this encoding. The probability of DETECTing the plane as being in an adjacent zone to the one it actually is in is a function of how close the plane is to the border between zones: the closer it is, the greater the probability of error. The student's ability to make accurate DETECTions in a given zone increases with training, i.e., the student is more and more likely to make an accurate DETECTion of a plane at a given distance from a border. This increase in ability is a function of the resources allocated to detection and the number of times the student has made a DETECTion in the particular zone. The time it takes the student to do a DETECT decreases with training. This decrease is a function of the resources allocated to DETECT and the number of times the frame is selected. Note that doing a DETECT of any zone reduces the time needed to do a DETECT of all other zones, whereas DETECTion accuracy improvement is specific to a particular zone, through experience with DETECTing that zone.

PCOMP AND THE SKB. The SKB, outlined in Table 2, is the structure for the PCOMP process. PCOMP directs the search through the SKB in order to determine the proper verbal encoding of the spatial information made available by DETECT.

The three levels of the SKB hierarchy can be viewed as three successive levels of resolution: "MAIN" having the least resolution and ABOVE/SLIGHT ("AS") or BELOW/SLIGHT ("BS") having the greatest. The arrangement is analogous to a hierarchy of maps in an atlas. MAIN encompasses the whole country, but only shows state boundaries for "ABOVE", "BELOW", and a major city -- "On Glidepath". However it points to where the next level of resolution can be found. ABOVE, for example, encompasses less territory but now brings the county of AS into view and denotes another city -- "Well Above" (glidepath). AS encompasses the least territory of all but finally brings into view the last two cities -- "Above" and "Slightly Above" (glidepath). The pointer "ACTUAL" signals ATTEND that encoding is completed and that an actual advisory is ready to be made. Note that in our implementation, the simulated student is assumed to have "read the manual," i.e., the student has already passed through the first schemata formation stage described by Rumelhart and Norman (1976) and Norman (1978).

Admittedly this specific hierarchical arrangement was intuitively designed, but we believe it is theoretically reasonable. These levels of resolution are congruent with Craik and Lockhart's (1972) concepts of levels of processing. The hierarchy evolved from an introspective observation of the decision tree used to compute the appropriate advisory. First, the On Glidepath advisory seemed unique in nature: it was the ideal elevation, and is the only elevation neither above or below the glidepath. If the plane is not on glidepath then it must be above or below it: this is the MAIN frame. Once we have decided the plane is above the glidepath, for example, then the next position which seems most easily delimited is the one at the end of the target -- Well Above. If the cursor does not cross the target at the end then the elevation must be one of the remaining two: this is the ABOVE frame. The last frame, AS, represents the final determination: is the target Above or only Slightly Above the glidepath?

We have previously alluded to the fact that the data structure evolves from this hierarchical construction to a single homogeneous frame. In the hierarchical construction only the highest resolution frames AS and BS are homogeneous, i.e., the quality of the data in each element of the frame is the same. In the MAIN frame for example the

On Glidepath element represents the code for an actual advisory, while the other elements are merely pointers to other frames. In our design two frames will merge when the learning level of all of the elements of the higher resolution frame reach a criterion level. When a merge takes place each element of the lower resolution frame is replaced by the corresponding element of the higher resolution frame. Thus, eventually, the MAIN frame will consist of seven elements each of them representing actual advisories. The learning value of each element of a frame is incremented each time that element is accessed. This incrementation is a function of the resources allocated to PCOMP and the number of times the element has been accessed.

It is PCOMP that directs the search and evolution of the SKB. Some frame of the SKB is always available to PCOMP. PCOMP directs three basic actions: (1) it determines the next SKB frame to be searched, (2) it increments the learning parameter associated with a particular SKB frame element, and (3) it directs the merging of SKB frames.

The essential function of a single PCOMP cognitive act is to determine the next SKB to be searched in order to find the appropriate verbal encoding of the spatial code made available by DETECT. This determination is a function of the zone DETECTed and the SKB frame which is currently available to PCOMP. PCOMP first determines if the frame is sufficient, i.e., if a verbal encoding element corresponding to the spatial code is even present in the frame. If one is not, we could say that the plane has moved outside of the student's current frame of reference. Therefore, PCOMP directs that the next search should begin again at MAIN frame.

If a corresponding verbal element is found in the frame, a number of actions are taken. First, the learning parameter associated with the element is incremented. Second, PCOMP determines if a merge is warranted. If it is, the merge is completed. Next, PCOMP examines the corresponding verbal element. If it is a pointer to another frame, this pointer is stored in short term memory. If the element is the encoding of an actual advisory, then the advisory is stored in short term memory and a flag is set indicating to ATTEND that an advisory has been encoded and no more PCOMPing is necessary unless the plane is DETECTed as having moved to another spatial zone.

The time needed for doing a PCOMP for a particular spatial zone decreases with practice in processing that particular zone. This decrement is a function of the learning parameter associated with the verbal element corresponding to the spatial zone.

This implementation of PCOMP encompasses the qualitative and quantitative changes in cognitive processes and data-structures outlined in the Section II. We have described the evolution of the SKB, and the fact that PCOMP has been programmed to increase speed with practice. In addition, the interaction of the PCOMP process with the evolving SKB yields a difference between the novice and the expert. Chatfield et al. (1979) noted that one capability of the expert that seems to facilitate processing is the expert's apparent ability to anticipate the movements of the target. This ability has been implemented in INSTRUCT's simulated student through the interaction of the PCOMP process and the evolving SKB. When the SKB is hierarchical the student's ability to "anticipate" is limited. The student can deal proficiently with a plane that changes zones as long as it stays in the student's frame of reference. The problem arises from the fact that the student's frame of reference unfortunately grows narrower as verbal encoding progresses. Because time is also progressing, the probability of a zone shift concomitantly increases. However, the merging of the SKB frames allows verbal encoding to be completed in a frame with broader reference. Therefore, an expert with a fully homogeneous MAIN frame: (1) only has to make one PCOMP (of MAIN frame) to verbally encode the planes spatial position, and (2) the plane will never drift out of the frame of reference.

CALL. This frame directs the student's output of an overt response. CALL may be selected by ATTEND whether or not the target's glidepath elevation is completely encoded. If encoding is complete, CALL prepares to output the corresponding verbalization. If encoding is incomplete, then CALL prepares an "educated" guess. In this case CALL will access the SKB frame in short term memory and prepare a verbal response that corresponds to one of the positions, randomly selected, within its frame of reference. Therefore, the further along the verbal encoding is, the greater the probability of the guess being correct.

Error may occur in the actual output of the verbal response. Ill-phrasing or speech degradation may occur with some probability on any verbal response. The probability of either occurring is much lower for a prepared advisory than for a guess. The probability of degradation to either type of advisory decreases with practice. This decrement is a function of the resources allocated to CALL and the number of times CALL has been selected.

CCOMP. This frame directs the student's use of internal feedback. The student can not inherently know if an advisory is correct or not. INSTRUCT's student does know: (1) if an advisory was missed, (2) if an advisory was made before final verbal encoding was completed, i.e., a guess, albeit a possibly educated guess, (3) if an advisory was the result of complete encoding. The student inherently acts to decrease the likelihood of the first two types of advisory each time they are encountered. This is done by attempting to increase the amount of PCOMPing between advisories. To do this the student increases the range on either side of the midpoint between call points where no competing cognitive act is allowed. This range is decreased whenever a completely encoded advisory is made. The minimum limit of this range is zero, the maximum is the full distance between call points.

WAIT. Currently the student is directed to SCAN and DETECT while WAITing. This is a default schema. With the present task the most important thing the student can do while WAITing is SCAN and DETECT. If the plane remains in the same zone as that verbally encoded, WAITing continues. If the planes shifts zones, then WAIT directs the student to return to PCOMP.

THE COMPUTER RESIDENT AUTOMATED INSTRUCTOR

OVERVIEW. As we reported in our introduction, Chatfield et al. (1979) described the necessary components of such an instructor. These components included a knowledge base, in which procedural as well as factual knowledge would be represented; a model of the student; a performance measurement or monitor component; a diagnostic component capable of deducing the state of covert student student processing and knowledge levels from overt student behavior; and a curriculum driver which could use the student model to determine the "tutorial" function or adaptive task scenario that would optimize the student's learning. In addition it was noted that a natural language interface would be useful in providing the student with a means for initiating information inquiries, in allowing the system to refine its diagnosis of student problems by interrogating the student, and provide the human instructor with convenient access to the system's data base on student performance and its pedagogical rationale.

INSTRUCTOR KNOWLEDGE BASE (IKB). The IKB is the core of the CRAI, just as the human instructor's knowledge is the core of his/her instructional decision making. It is clear that a major element in the human instructor's knowledge base is a model, albeit possibly an intuitive model, of covert student psychological processes. Instructors, as well as students, refer to these subjective processes as if they are real. LSO's often refer to the student as "getting the eye," an obvious reference to what we formally defined as the perceptual processing stage. In addition, instructional strategies and alternatives reflect an implicit assumption of the existence of separate processing components. Instructors may prescribe special exercises to develop the student's detection skills, when they surmize that the student's perceptual abilities are weak. They may instruct the student to review the rules for developing advisories, when they suspect deficiencies in what we have defined as the mental computation processing stage. Thus, instructors' pedagogical behavior adds, at the least, face validity to our theoretical hypothesis of staged processing.

In addition, this behavior indicates the need to specify these processing stages in units which are pragmatic with respect to the training alternatives available, as well as being valid psychological constructs. Furthermore, these units must be meaningful and intuitively appealing to the student to facilitate diagnostic introspection and instructional influence.

The Basic Student Model. The IKB must then contain information about the student and the task and the relationship between them. In INSTRUCT all of this information is completely defined. In particular, we know the exact implementation of the psychological concepts of the nature of the student presented in Section II. For example, we know that initially mental computation of the verbal encoding of a target elevation operates on a three level hierarchical data-structure. We even know the exact form of the process by which this encoding takes place and the way in which the data-structure evolves. In the real world this precise information would not be available. Conceptually we would know that some form of mental computation takes place, but we would not know its exact form. We might even conclude that the process operates on some form of hierarchical data-structure, but we wouldn't know how many levels. In other words, given our present psychological knowledge, our model of the student might be conceptually, logically and/or pedagogically sufficient, but it will not be fully determined.

Thus, our representation of the student processes should not be as detailed as they could be. It would be more instructive to approach the representation of these processes from a more naive and general perspective, as we would have to do in an actual system design situation.

In the design of the simulated student, we chose to work with only a few basic processes which would adequately illustrate the training system design. We certainly wouldn't want greater complexity in our model of that student. Thus drawing on the same literature which underpins the simulated student we would discern a number of basic processes: attentional decision making, resource allocation, perceptual/scanning, detection, mental computation, and verbal production. With our present knowledge, we do not know the precise sequence these processes are brought into play to produce an elevation advisory. However, it would be reasonable to assume that the production of an advisory would require each of these processes, and logically, in a single action sequence, scanning would precede detection which precedes mental computation which precedes verbal production. Figure 8 illustrates this action sequence and its relationship to other processes. It would also seem reasonable that a failure in any of the links in this logical chain would cause the whole sequence to fail, or at least produce an erroneous response. Note also that only verbal production is overtly measurable, i.e., the state of the other processes must be inferred from the qualities of this verbal production.

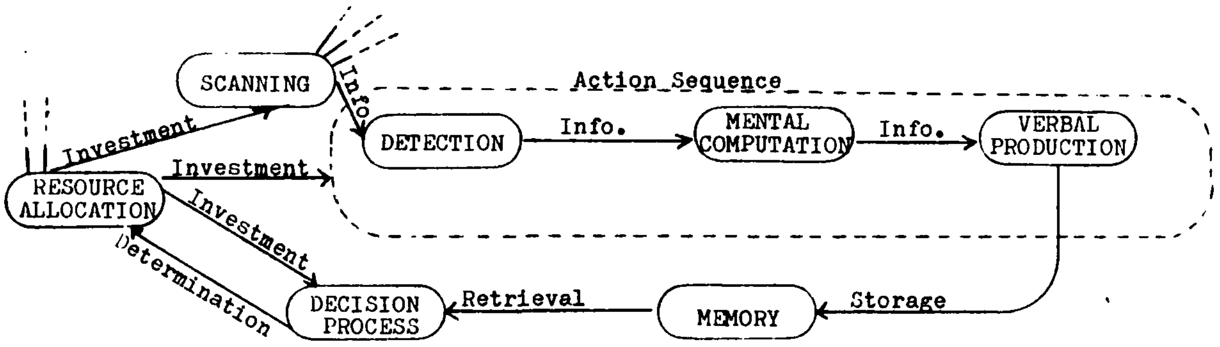


Figure 8. Relationships between basic processes.

Elaborating the Basic Model. Two problems now arise: (1) what pragmatic information about these processes in the basic model are necessary for artificially intelligent pedagogical decisions, and (2) how should this information be represented for use by the CRAI.

Early in the current contract effort we examined the pedagogical significance of actually being able to estimate the resource allocation parameters relevant to the momentary acquistion rate. We developed a computer simulation model which embodied and allowed us to manipulate the cognitive resource allocation and resource allocation acquisition characteristics of an exemplar processing component.

The simulation involved a number of assumptions. First, that the student would be required to master two independent cognitive components of a task. Second, the student would operate under limited resources: the resource allocations would add to a constant, and manipulation of allocations was limited to a swap of resources between the two components. Third, mastery of these components would be acquired through practice. Fourth, that for a given resource allocation for a given task, the relationship between the level of mastery acquired and the amount of practice trials completed would be described by a learning curve which was continuous and monotonic, and concave downward. Fifth, the rate at which the learning curve would approach its asymptotic momentary acquistion rate would be positively related to the level of cognitive resources allocated to the learning of a particular component throughout each practice trial. However, the exact nature of that relationship would be determined by the resource allocation acquisition function (RAAF) for each cognitive component. A detailed explanation of the RAAF can be found in Chatfield et al. (1979).

With this model, various combinations of RAAF's for the two cognitive components could be established. We could then examine the interaction effect of the combinations with various resource allocations on the optimum pedagogical strategy (that strategy which achieved the greatest increment in learning on each trial). For example, we were able to examine the effect of equal allocation to both components and various degrees of differential allocations, crossed with various learning curves.

We examined the effect on trials to acquistion, and in particular the pattern of reallocation resources yielded by the optimization constraint. Tables 3 and 4 illustrate the typical acquisition pattern. In each example the RAAF's for the two tasks were the same. The same RAAF's were also used in both examples. Note that the higher differential allocation yields a higher rate of learning. This finding held under all combinations of RAAF's. However, note that the pattern of reallocation is the same in both examples. This general pattern also held for all combinations of RAAF's and allocations. The pattern initiates by allocating the greatest resources to the task with the lower average momentary acquisition rate (the average momentary acquistion rate is a function of the RAAF. If the RAAF's are equal the initial allocation is arbitrary). The allocations are then swapped until learning on the remaining task exceeds that of the first. Allocations are then swapped again. The pattern continues, and the point of reallocation is essentially always that point when the task with the higher allocation exceeds the learning of the remaining task.

The results of this simulation indicate that a precise determination of RAAF's probably would not add significantly to the pedagogical knowledge base, i.e., its computational cost would not be offset by its pedagogical benefit. Even an error made in the initial allocation, by not knowing the relative RAAF's, would not make a significant difference in overall acquisition rate. In addition, regardless of the RAAF the pedagogical strategy of choice is to manipulate the student into as great of a differential allocation as

TABLE 3. TURNPIKE SIMULATION WITH HIGH DIFFERENTIAL

TRIAL	RESOURCE TASK 1	RESOURCE TASK 2	LEARNING TASK 1	LEARNING TASK 2	TOTAL LEARNING
1	.9	.1	.478297	1.00000E-07	.478297
2	.1	.9	.478297	.478297	.956594
3	.1	.9	.478297	.727826	1.20612
4	.9	.1	.727826	.727826	1.45565
5	.9	.1	.858006	.727826	1.58583
6	.1	.9	.858006	.858006	1.71601
7	.1	.9	.858006	.925921	1.78393
8	.9	.1	.925921	.925921	1.85184
9	.9	.1	.961353	.925921	1.88727
10	.1	.9	.961353	.961353	1.92271
11	.1	.9	.961353	.979838	1.94119
12	.9	.1	.979838	.979838	1.95968
13	.9	.1	.989481	.979838	1.96932
14	.1	.9	.989481	.989481	1.97896
15	.1	.9	.989481	.994512	1.98399
16	.9	.1	.994512	.994512	1.98902
17	.9	.1	.997137	.994512	1.99165
18	.1	.9	.997137	.997137	1.99427
19	.1	.9	.997137	.998506	1.99564
20	.9	.1	.998506	.998506	1.99701
21	.9	.1	.999221	.998506	1.99773
22	.1	.9	.999221	.999221	1.99844
23	.1	.9	.999221	.999594	1.99881
24	.9	.1	.999594	.999594	1.99919
25	.9	.1	.999788	.999594	1.99938
26	.1	.9	.999788	.999788	1.99958
27	.1	.9	.999788	.999889	1.99968
28	.9	.1	.999889	.999889	1.99978
29	.9	.1	.999942	.999889	1.99983
30	.1	.9	.999942	.999942	1.99988

TABLE 4. TURNPIKE SIMULATION WITH LOW DIFFERENTIAL

TRIAL	RESOURCE TASK 1	RESOURCE TASK 2	LEARNING TASK 1	LEARNING TASK 2	TOTAL LEARNING
1	.6	.4	.0279936	1.63840E-03	.029632
2	.4	.6	.0295861	.0295861	.0591722
3	.4	.6	.031176	.0567515	.0879275
4	.6	.4	.0582969	.0582969	.116594
5	.6	.4	.0846586	.0598398	.144498
6	.4	.6	.0861583	.0861583	.172317
7	.4	.6	.0876555	.11174	.199395
8	.6	.4	.113195	.113195	.226391
9	.6	.4	.13802	.114648	.252668
10	.4	.6	.139432	.139432	.278865
11	.4	.6	.140842	.163523	.304365
12	.6	.4	.164893	.164893	.329787
13	.6	.4	.188271	.166262	.354532
14	.4	.6	.189601	.189601	.379202
15	.4	.6	.190929	.212287	.403215
16	.6	.4	.213577	.213577	.427155
17	.6	.4	.235592	.214866	.450458
18	.4	.6	.236845	.236845	.473689
19	.4	.6	.238095	.258208	.496303
20	.6	.4	.259423	.259423	.518847
21	.6	.4	.280155	.260637	.540792
22	.4	.6	.281334	.281334	.562668
23	.4	.6	.282512	.301452	.583964
24	.6	.4	.302597	.302597	.605193
25	.6	.4	.32212	.303739	.625859
26	.4	.6	.32323	.32323	.64646
27	.4	.6	.324339	.342175	.666514
28	.6	.4	.343253	.343253	.686506
29	.6	.4	.361638	.344329	.705967
30	.4	.6	.362684	.362684	.725367

possible. The reallocation decision would then depend on accurately discerning the students relative learning levels. It became apparent that this discernment could be more efficiently achieved through heuristic (as opposed to mathematical optimization) techniques in the voice-based task environment, i.e., through the use of artificial intelligence deductive techniques. To facilitate this process, it would be necessary to elaborate the basic model.

To continue developing a representation system, the generic form of a process needs to be examined more closely. Figure 9 shows what might be considered the general form of an information processing unit. The figure depicts an unnamed process which would receive information from another process in a preceding stage, process the information and pass the information to a subsequent processing unit (unless it itself is the termination of a chain).

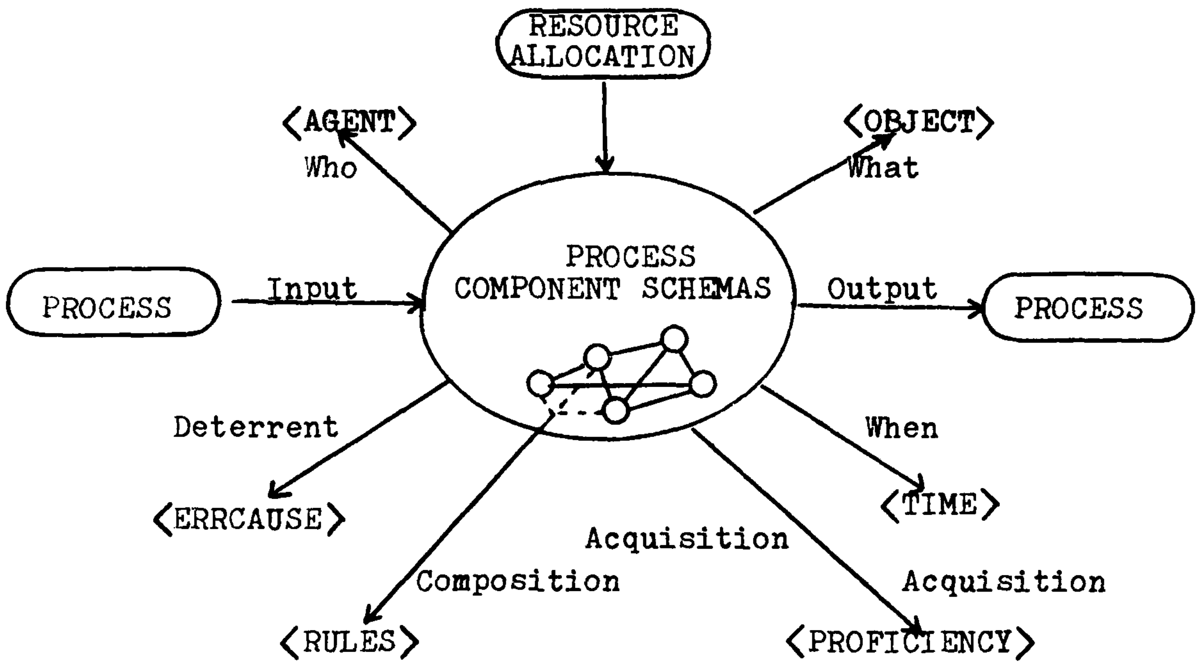


Figure 9. The generic form of a basic process.

The process could be thought of as being composed of component schemata, which may or may not be known to an instructor or possibly even the student. These component schemata represent the knowledge structure or rules by which the process operates. At the beginning of training or during remediation, the tutorial function may establish these rules. For example, in computing a trend call the student may be told to consider the current position of the aircraft and whether it is approaching or departing from the glidepath.

The exact rule could be stated in the form of a series of "If-then" statements that the student would try to remember and use in processing. The structures may evolve to more efficient forms during training. For example, the If-then statements ultimately could be replaced by simple associations. Thus, the instructor and possibly even the student may not be aware of the exact state of the evolution of the component schemata during training. For this reason, the schemata are left in a non-specific form at this point in our representation.

As also can be seen in Figure 9, conceptual information regarding processes is *included* in our generic representation, mainly for the benefit of our interrogatory interface. As noted, an act of information processing would include an actor (an agent who does the processing), a direct or indirect object (that which is being processed), and a time in which the act occurred (usually expressed in reference to the location of the aircraft at the time).

With this information, the activation of a processing unit (which we will denote as a PROC) can be expressed as a predicate, such as:

PROC(AGENT, OBJECT, TIME)

after the manner described in Norman, Rumelhart and the LNR Research Group (1975).

A particular instantiation of the predicate could be:

DETECT (STUDENT, WAGP, 2 MI MARK)

which depicts the idea that the student detects the aircraft as well-above the glidepath at the two-mile mark. With the arguments filled in, the predicate now becomes a proposition with a truth value, i.e., the student either did or did not detect the aircraft as stated. The determination of the truth value becomes the purview of the diagnostics module in INSTRUCT with the help of the interrogatory interface.

It was determined that other information needed to be represented in our generic depiction of a PROC. In the diagnostics routines, searches would be made for the causes of an incomplete or incorrect functioning of a process, i.e., some particularly conceptual entity (ERRCAUSE) would be responsible for impeding the progress of the PROC toward a correct and timely conclusion. Further, reference would need to be made to the various rules comprising the initial structural base (component schemata) of the PROC, as well as some form of current proficiency information on the PROC, which would be used by the student model.

An example of a mental computation PROC (which we will refer to as an MCOMP) is shown in Figure 10. Using the information in the diagram, a frame-like data structure can be created which will carry the information required by INSTRUCT.

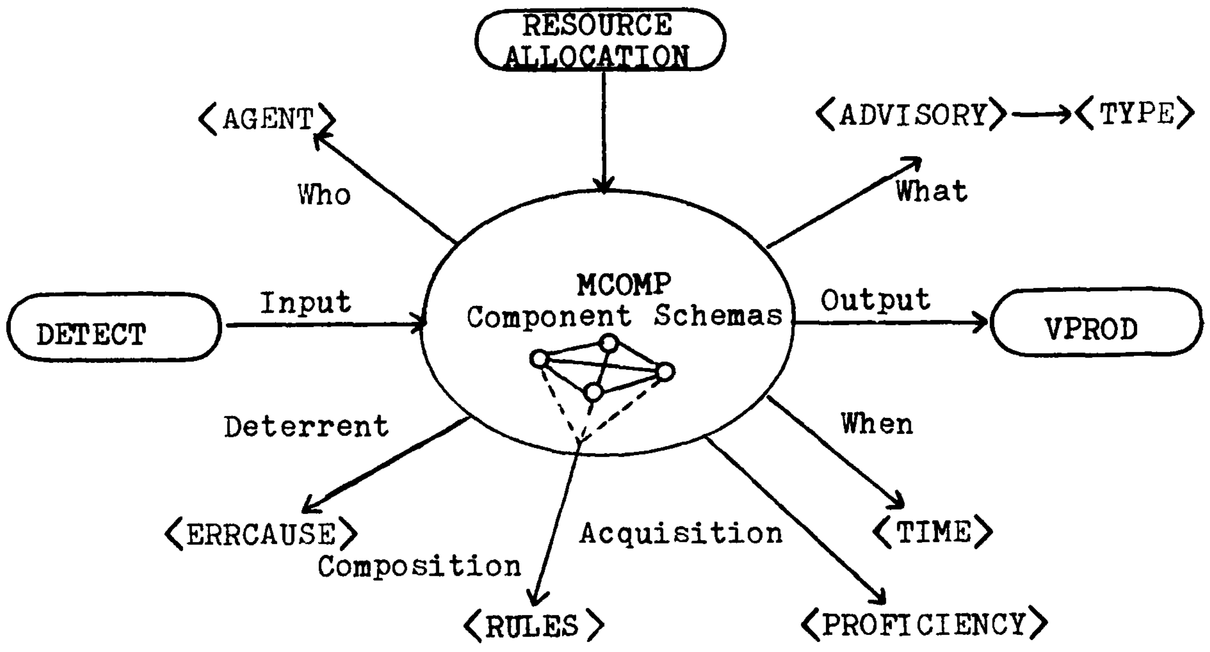


Figure 10. Representation of a mental computation PROC.

An example is shown in Figure 11. As can be seen, the array is organized into rows, which could represent the various slots in a frame, which need to be filled. The first element in a row contains the slot label, such as "SUBJECT." The second element contains the slot entry or value itself, for example, "(STUDENT)," which would be the actor in an MCOMP action. The third element is the constraint or condition for value assignment.

MCOMP

TYPE	?	
INPUT	DECTECT	
OUTPUT	VPROC	
SUBJECT	(STUDENT)	AGENT
VERB	(\THINK)	MCOMP
OBJECT	?	CALL
DETERRENT	#16	ERRCAUSE

Figure 11. Individual data-structure for representing the PROC MCOMP.

In the "SUBJECT" row, the value inserted must be a noun of the "AGENT" case. The left-most column is fixed and represents the attributes always present and needed to understand the action of an MCOMP. The middle column is to be filled during the process of instantiation. It may, however, contain default values that are used in the absence of overriding information. In INSTRUCT, the default values were denoted by parentheses. The question marks denoted the fact that no default values were to be found and that it was mandatory for the system to seek the appropriate values. In some cases, the entry was blank indicating again no default value, but also that it was not mandatory that an entry be obtained. The number in the last entry represents a pointer to another data structure that ultimately would fill the "DETERRENT" slot. Additional detail concerning the meaning of some of the entries will be explained more fully in our discussion of the interrogatory interface.

The six PROCs shown in Figure 10 all were represented by data structures similar to that shown in Figure 11. Considering the numerous instantiations possible, however, a far larger number of specific processing instances could be represented as variants of the basic six. But for completeness, more than just PROCs needed to be represented. The representation of the basic cognitive processing units would be sufficient for most of the INSTRUCT modules, such as the student model, curriculum driver, etc., but not for an interrogatory interface. The interface would require a small semantic network consisting of the concepts required for discourse with the student. Figure 12 shows a small portion of the network that would support such a discourse.

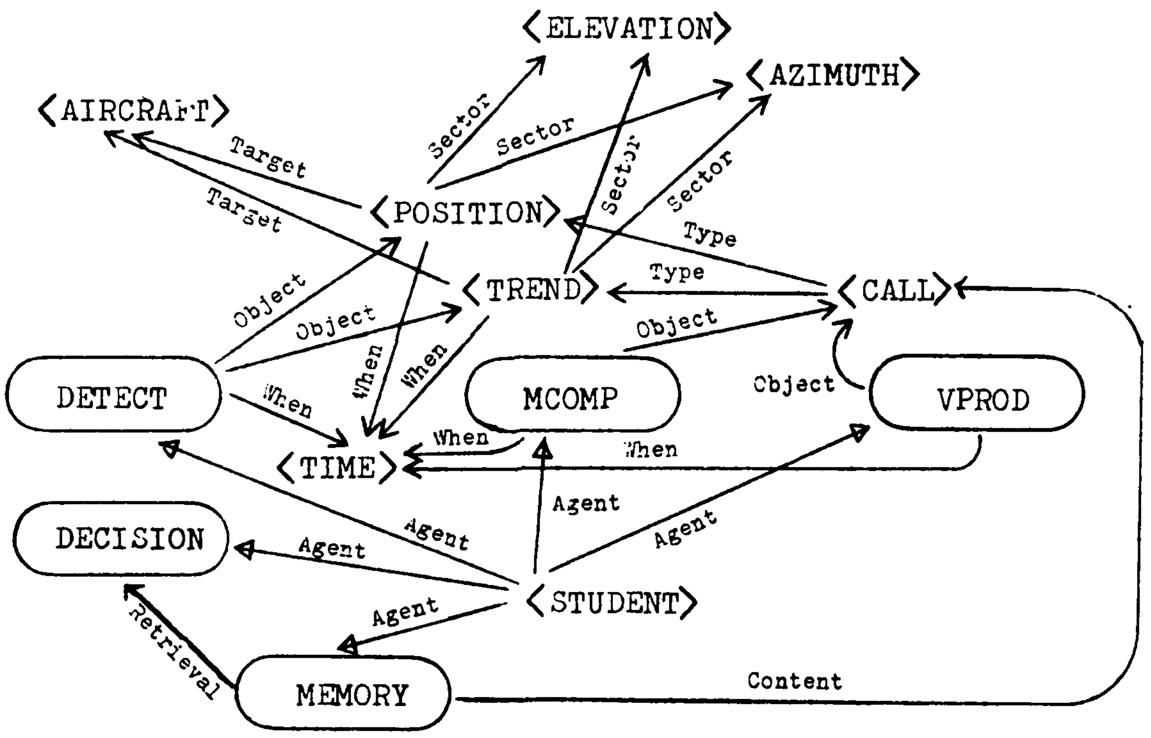


Figure 12. A portion of the type of semantic network that would be required to support discourse regarding the basic processes.

As might be suspected, the network could get to be quite large unless constrained. As stated previously, intelligent systems need a rather limited domain in which to operate. INSTRUCT's domain is even more limited as its only purpose was to devise a model to show feasibility, not to create an extensive operational system. Thus, in keeping INSTRUCT's domain limited, the only nodes that were included were those that were needed to carry on a dialogue about the student's performance. The data structures for other verbs, some of which are shown in Figure 13, were created. Verbs such as these would give INSTRUCT the capability of discussing "the occurrence of events," "memory," "distracting events," etc. The other nodes in the network, such as nouns, were represented lexically.

```
          .
          .
          .
OCCURRENCE
     SUBJECT          ?             EVENT
     VERB                         OCCURENCE
     PREPV      (\THIS TIME)         TIME

STATIVE
     SUBJECT          ?
     VERB           (IS)            STATIVE
     OBJECT           ?

RETRIEVE
     SUBJECT      (STUDENT)          AGENT
     VERB         (REMEMBER)        RETRIEVE
     OBJECT           ?
          .
          .
          .
```

Figure 13. Data structures for verbs showing default values.

In short, representation in INSTRUCT centered around basic cognitive processing primitives. It was assumed that the variants of task scenarios and performances would simply involve instantiations of the basic processes. Information and pointers regarding these PROCS were represented by frame-like data structures. Additional frames were developed around other verbs that would be needed for understanding discourse with the students. Other concepts required for discourse were simply represented as lexical entries. In the subsection on the interrogatory interface, we will explain in greater detail how these representations are actually used.

DIAGNOSTICS. This component of the CRAI directs the process of inferring the state of the student's covert cognitive components from the student's overt responses. The subprocesses involved fall into categories analogous to those outlined for the student. The monitor is the CRAI's feature detector or sensory surface. This information is first passed through a preliminary encoding which identifies what surface events are indicated by the monitor record. A secondary encoding then interprets the pattern of events in terms of possible underlying covert causes. These causal hypotheses may be validated by directing the request of new information from the environment through an interrogatory interface.

In our implementation of INSTRUCT we chose to use a passive monitor of the student's overt reponses during the task. This monitor is analogous to a simple tape recorder. The monitor records the planes vertical and horizontal position at the time of the response as well as the response itself. We relegated even preliminary interpretation of surface events (e.g., a wrong advisory) to take place between practice approaches. In a real-world system some efficiency might be gained by having this work done during the relatively long pauses (for a computer) between student reponses. Recall, that in our case this time is filled with recording the simulated student's cognitive actions. In addition, we thought it would make the CRAI conceptually more coherent to schedule all of its encoding routines together. In any case the techniques we implemented are representative of those which might prove useful in the real world, even should the scheduling of their use might be different.

Preliminary Diagnosis. Quick intuitive analysis might lead one to believe that a "late advisory" or a "speech degraded advisory" are simple events, but careful introspection will reveal that they are indeed conceptual interpretations of the events occurring in the context of the task environment. This context provides us with constraints on the interpretation of these events which allow us to efficiently encode most of them. This use of constraints is a powerful AI technique. It is used in computer visual interpretation (Waltz, 1975) and in natural language processing (see

Winston, 1979, for a review). Indeed one of the major difficulties in natural language processing is defining the constraints of natural language. This has led, on the one hand, to the use of "programming" languages when talking to computers, because these languages are designed with rigid constraints. On the other hand, research continues into defining the algorithms needed to provide the computer with the capabilities to "discover by example" these constraints (see Carbonell & Michalsk, 1981, for a review of current research).

In its preliminary encoding, the CRAI looks for both evidence of student initiated events (e.g., a late advisory) and environmental contingencies which might contribute to student difficulties (e.g., the plane crossing a zone boundary just prior to a call point). The constraints for the interpretation of such events are programmed into a decision tree. For example, determine if an advisory was missed. The positions at which advisories are supposed to be made are predetermined. Second, the simulated student never makes an early advisory. Starting at a given call point, the CRAI applies the rule: if no advisory was made between this call point and the next, then an advisory was missed. The task environment constraints also determine the hierarchy of the decision tree. For example, if an advisory is interpreted as missed, obviously no other student initiated events regarding that call exist. Therefore, further examination for such events can be terminated. Efficiency is facilitated by eliminating redundant processing.

This preliminary encoding is carried out until the relevant events surrounding each call point are as fully determined as possible, and encoded in a form which facilitates further diagnosis. Some events can not be fully determined at this level. For example the CRAI does not contain the necessary constraints to determine if an advisory was speech degraded or simply improperly phrased. The CRAI has constraints to determine that the observed advisory does not match a model of the expected advisory, and that it does not match the a model of any other proper advisory. In other words, the CRAI knows what the advisory is not. As will be shown below, even this much information is inferentially useful. However, full determination of the event would take a considerable amount of knowledge about natural spoken language constraints which may not be available. Such an instance is one case where the interrogatory interface would prove useful (the details of that procedure will be outlined in our discussion of the interrogatory interface, later in this section).

Secondary Diagnosis. Our conceptual model of the student would suggest that certain patterns of surface events might be associated with specific covert problems. For example, an inefficient detection component would be expected to yield a missed or late advisory because of its slowness or a wrong advisory because of its inaccuracy. It would be less directly associated with speech degradation or phrasing problems. On the other hand an inefficient verbal production unit would be directly associated with degraded speech and phrasing problems, but would not be expected to vield an advisory for the wrong elevation. The pairing of a pattern of events with a causal assertion is an example of a production rule.

The CRAI contains a production system which we have called the Causes-Event data-structure (CEDS). This production system defines the expected events associated with inefficiency in each of the cognitive processes, as well as with external contingencies, e.g., a late elevation zone boundary crossing. By examining the match between the observed events with each production rule's pattern, the CRAI can establish the probability that the observed events were caused by the associated inefficiency or contingency. One examination we shall make will scrutinize the CRAI's ability to infer covert causes from the student's overt responses.

Although a unique pattern of events can be associated with each possible cause, two factors hinder positive causal identification at this level: (1) not all events associated with a cause will necessarily occur because that cause is present, and (2) there is considerable overlap between the patterns of events associated with different causes. For example, an inefficient mental computation process would be associated with a missed or late advisory because of its slowness. This slowness might also force the student to guess, yielding an advisory for the wrong elevation. This guessing might also be manifested in degraded speech or incorrect phrasing. Now let's assume the observed events are a late advisory for the wrong elevation. These two events are associated with an inefficient detection component, as well as the inefficient mental computation. This is another example where it would be useful for the CRAI to return to the environment (i.e., the student) in an attempt to further verify its hypotheses concerning the state of the student's covert cognitive processes.

Interrogatory Interface. Creating a natural language interface was never the intent of the present project. Many such interfaces have been built and could be adapted to the needs of an actual voice-based trainer. Much is known about the process of understanding natural language and even though much has yet to be developed, our efforts were not designed to add to that knowledge. Those wanting to know more about the application of natural language in training could consult Burton and Brown (1979).

As was noted in the previous subsection, there was a real need to provide the capabilities for discourse with the student for diagnostic purposes. Furthermore, this likely could be the case in most training situations. We, therefore, sought to add these capabilities in some small amount to provide an example of its usage and to augment the diagnostic component of INSTRUCT. However, since creating an extensive natural language interface was not the purpose, a smaller version was created and was more modestly termed an "Interrogatory Interface." Its purpose was to serve only specific diagnostic needs (thus leading to the use of certain shortcuts) while demonstrating some of the techniques and capabilities of a larger interface.

The interface was composed of three major sections: the parser, the understander and the production unit. All three components make use of the frame-like data base (discussed in the subsection on elaborating the basic model) and a lexicon. The parser's function was to accept the student's input, parse the input into its syntactic units and then pass the information on to the understander. The understander then would attempt to understand the input and act upon it. The action taken may be in the form of the generation of a command or a question. If the understander was to produce another verbal response, it passed the appropriate information on to the production unit which transformed it into a syntactically correct verbal response. The subsections that follow will describe each of the interface components in more detail.

Parser. The parser was made up of two components: a syntactic specialist and a sentence specialist. The syntactic specialist made use of techniques and notations which were closely related to the Augmented Transition Network (ATN) formalism developed for natural language grammars. Those interested in reading an early comprehensive discussion of an "Augmented Finite State Transition Network" should consult Woods (1970). Other sources are Simmons (1973) and Norman and Rumelhart (1975).

A transition network, simply stated, is a formalism for describing the rules by which a process makes a transition from state-to-state. In this case, the states are syntactic units or words, and the rules for transition are given by grammatical constraints. Figure 14 shows a simple transition net for a noun phrase in the parser.

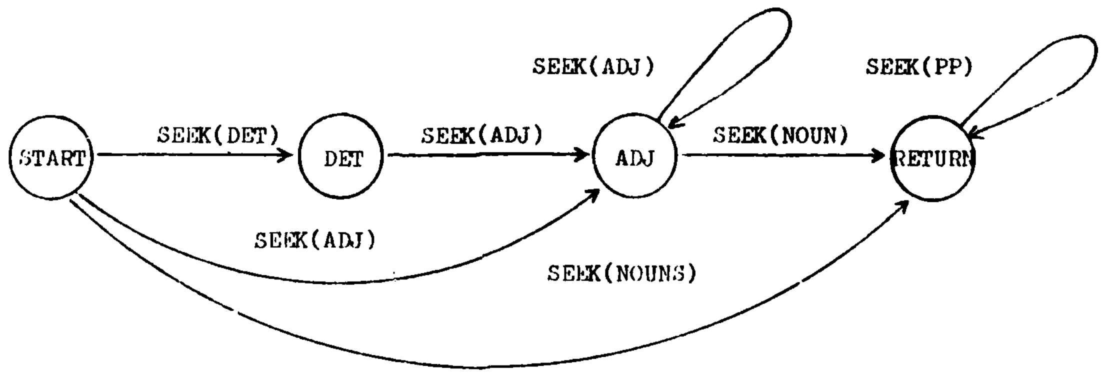


Figure 14. Simple transition net for a noun phrase.

The circles or nodes represent the states, while the arcs connecting the nodes represent the means by which the process can affect a transition. The term "SEEK" in the current case refers to a function whereby the system checks the appropriate portion of the lexicon (the internal dictionary) as specified by the argument for the word in question. Where nodes have multiple arcs or routes by which the process may leave, the arcs or SEEK functions are tried and executed in clockwise order until a successful transition is made.

As an example, suppose the student made reference to:

THE DUMB PILOT.

Starting at the first node, the system would check the list of determiners to see if the entry "THE" was a member. If it was, then transition to the next state labeled "DET" was made which indicated that a determiner was identified. Had the word "THE" not been found in the determiner list, the system would have then checked the adjective list, followed by the noun list.

Having just found the determiner, however, the adjective list now is searched for the second word, "DUMB." Eventually the process makes its way through the net placing the words in their appropriate registers (Figure 15) for later consideration by the sentence specialist.

An extensive ATN also would take notes as it progresses from node to node to determine the features or properties of the phrase. For example, when seeking the determiner it would take note of whether it was definite or indefinite and its number (singular or plural). Assuming the adjective and noun were in agreement, these features would become properties of a noun phrase. Because the sentences we were to parse in the current context were to be quite simple, some of the more extensive "note-taking" was omitted as one of the shortcuts.

DET	:	THE
ADJ	:	DUMB
NOUN	:	PILOT

Figure 15. The result of the parsing of a noun phrase in which the words are sorted into their appropriate registers.

Once the transition net for a noun phrase was obtained, it could be defined as a function, such as "SEEK(NP)." This function could then be used in a transition net for prepositional phrases, as shown in Figure 16.

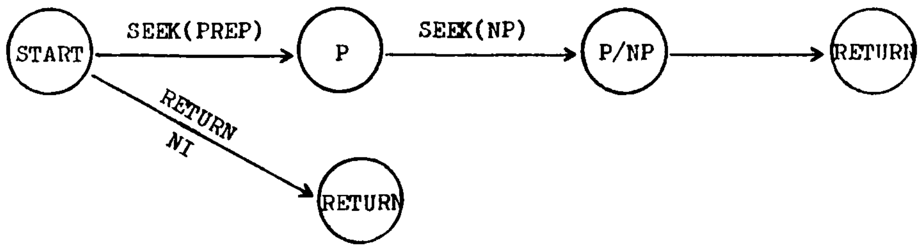


Figure 16: Transition net for a prepositional phrase.

Note that the procedure checks for a preposition first, followed by the search for a complete noun phrase. Denoting the function as "SEEK(PP)," it must then be added to the noun phrase function as shown on the last node in Figure 14. Thus, within the NP function, the PP function is called, which itself calls the NP function.

This recursive use of the functions allows the parsing of noun phrases with considerable imbedding of prepositional phrases, such as:

THE TIRED PILOT IN THE PLANE AT THE 2-MILE MARK

A function was similarly developed to parse a verb phrase. This routine not only would identify the verbs and adverbs, but also would take note of tense, person, number and whether the verb was in progressive form. Determination of the characteristics of the verb had to be supplemented, of course, with information obtained from an auxiliary verb, if there was one. Similar to the noun phrase function, any verb phrase identified was followed by a search for a modifying prepositional phrase.

With the collection functions searching for particular phrases, a larger net for complete sentences could be assembled as shown in Figure 17. This particular net was designed for a simple declarative sentence in active voice.

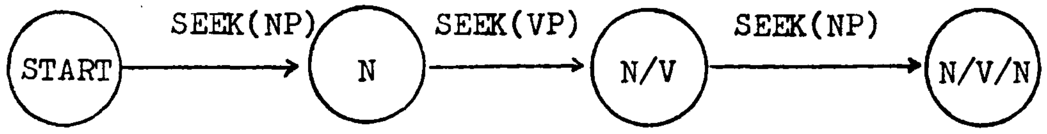


Figure 17. Transition net for a simple declarative statement.

A net for passive voice was somewhat more complex, as shown in Figure 18.

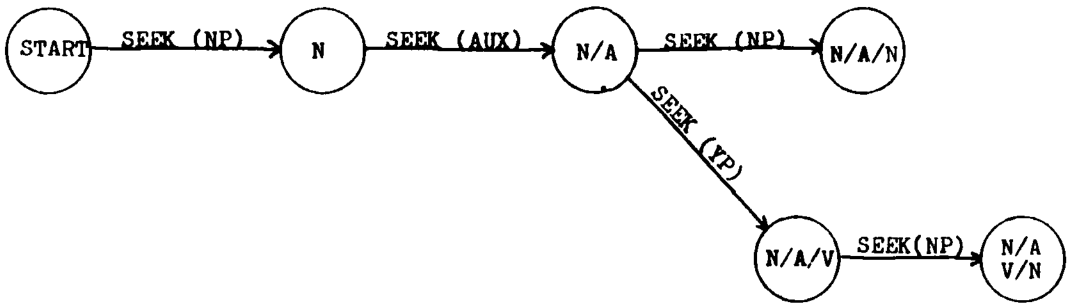


Figure 18. Transition net for declarative sentences in passive voice.

The two nets together make possible the parsing of sentences, such as:

THE PLANE PASSED THE 5-MILE MARK. (active)
THE 5-MILE MARK WAS PASSED BY THE PLANE. (passive)

The parsing of the first sentence can be seen quite readily. The second sentence begins with a noun phrase, followed by an auxiliary verb (was) and a main verb (passed). The verb phrase was completed by the introduction of the prepositional phrase "by the plane" (the SEEK(VP) routine contains a SEEK(PP) routine). At that point, the sentence ended prior to the identification of another noun phrase. It should be noted that the parsing process could exit from any of the states in the net if it did not have additional phrases to parse.

At this point, it can be seen that an extensive series of transition nets could be assembled to create a large parser. As our purpose was only to create a small demonstration interface, we chose to limit the inputs to simple sentences with only a single verb phrase. Should the student's input take the form:

I CANNOT REMEMBER WHEN I NOTICED THE DEVIATION,

only the first clause would be used, the second being ignored. It was felt that the exclusion of subordinate clauses, compound sentences, etc., would simplify the design of the interpreter, but still allow the interface to function sufficiently well to be an aid to the diagnostics module.

In addition to the transition nets for declarative sentences, nets for imperative, Wh-interrogatory and yes-no interrogatory sentences were added.

Thus, the parser now would handle such sentences as:

STOP THE PLANE!
WHAT SHOULD I DO NOW?
WAS MY CALL CORRECT?

By these different sentence forms, however, it must be remembered that the transition nets described simply identified each of the words as to whether they were nouns, verbs, prepositions, etc. Thus, the transition nets constituted the word specialist portion of the parser. Each time a word was identified as a certain part of speech, it was tagged as shown in Figure 13, and then written into a portion of memory called the parse blackboard. The term blackboard was used to refer to information that might periodically be modified and used by other modules.

The task of the sentence specialist was to identify the syntactic groupings of the words within the sentence. For most of the simple sentences that the interface would encounter, the task was not difficult. Usually the noun phrase preceding the verb phrase could be counted upon to be the subject, with the noun phrase following the verb to have the object.

However, an intransitive sentence does not have a direct object, as in this example:

I WAS DISTRACTED.

Additionally, the sentence could be in passive voice, such as:

I WAS CONFUSED BY THE MOVEMENT OF THE PLANE.

rather than in the active voice, as in:

THE MOVEMENT OF THE PLANE CONFUSED ME.

In some cases, the subject may be an interrogatory pronoun, as in:

WHO IS SPEAKING NOW?

For these reasons and others, a sentence specialist was needed to locate the subject noun, verb and object noun (if any). It did this basically by noting the order of the phrases in the sentence and then classifying the sentence by type and voice so that the appropriate phrases would be identified. Once the identification was complete, the subject, object, etc., were also noted on the parse blackboard.

Production Unit. The production unit was the module responsible for generating verbal responses in the system's dialogue with the student or human instructor. It was decided that we should explore sentence generaton, even though the current application was small enough that most responses from the system could have been of the script-based, preprogrammed type. The function of the sentence production unit was to take information passed from the understander, and transform it into a grammatically correct verbal response, just as a human's verbal response may begin with some sort of deep structure. An explanation of how the production unit functions will involve a description of the type of information it received from the understander, how it looked up words in the lexicon and how it transformed individual word units into a sentence.

When the system would have to make a verbal response to a user, student or human instructor, it would record and modify certain kernel thoughts on the blackboard (again an analogical reference to a modifiable bit of data used by several modules). This blackboard would contain references to a subject, object, verb and any modifying nouns, adjectives or adverbs. Figure 19 shows an example of such a list of entries.

STUDENT	Subject	Tense	Past
\TREND	Object	Number	Singular
DETECT	Verb	Person	Second
\NOT	Negative	Progression	Static
		S type	Y/N
		Destination	Student
		Source	System

Figure 19. Blackboard entries, in a form available for sentence production

Inherent in this information being passed to the production unit is the indication that the subject of the sentence is to be the student, the verb -- "DETECT," the object -- "TREND," and the verb is to have a negating modifier -- "NOT." The backslash preceding TREND was simply an internal code that the entry was to be used without any evaluation or modification (similar to the use of a single quote in LISP). In contrast, the verb would need modification, in this case, tense. The second series of entries pertained to the grammatical characteristics to be produced. As can be seen, the sentence was to be in the past tense, second person singular, and with the verb not of progressive form.

The type of sentence to be produced was a yes/no ("Y/N") interrogatory. Furthermore, the destination of the communication was the student, in this case also the subject of the sentence, with the system as the source.

To create the sentence, all unquoted entries (those without the backslash) would be evaluated. In the case of the subject, if the entry was the same as the destination, it would be replaced by the pronoun "you," but if the source were the subject, it would be replaced by "I." The evaluation of the verb, of course, would be that of determining tense.

In going to the lexicon in search of DETECT the various other tense forms also would be found, for example:

(DETECT DETECTS DETECTED DETECTING).

Because the sentence to be produced in the example was an interrogatory, the tense was to be indicated by an auxiliary verb, in this case "did." As could be expected, the tense evaluation was via a series of if-then rules based not only on the "tense" entry, but also on sentence type, progression, person and number.

Following the evaluation of the entries on the blackboard, the production unit first would assemble the words into phrases and then the phrases into a sentence using the logical equivalent of the rewrite rules shown in Figure 20.

1. S ⟶ NP(Subj) + VP + NP(Obj)
2. S ⟶ Aux + NP(Subj) + VP + NP(Obj) + ?
3. S ⟶ Wh- + Aux + NP(Subj) + VP + NP(Obj) + ?
4. NP(Subj) ⟶ Noun(Subj)
5. NP(Subj) ⟶ Adj(Subj) + NP(Subj)
6. NP(Subj) ⟶ Art + NP(Subj)
7. NP(Subj) ⟶ NP(Subj) + PP(Subj)
8. PP(Subj) ⟶ Prep + NP(PP-Subj)
9. VP ⟶ Verb
10. VP ⟶ Adv + VP
11. VP ⟶ VP = PP(Verb)

Rewrite rules used to construct the phrases where S=sentence, NP=noun phrase, VP=verb phrase, PP=prepositional phrase, AUX=auxiliary verb, and Wh- is a Wh- word (e.g. who). Rules for generation of NP(Obj) were similar to rules 4 - 8.

Figure 20. Rewrite rules.

These rules constituted a simple phrase structure grammar, sufficient for our use.

To make use of the rules in Figure 20, words not supplied directly by the understander would need to be looked up in the lexicon. For example, say that the understander had supplied

(PLANE(SUBJ)) and (3-MI MARK ((PPSUBJ)(LOC))

as words from which to construct a noun phrase. The first entry, as denoted, is to be used as the subject of the sentence, while the second entry, as denoted, is to be the object of a prepositional phrase, which in turn modifies the subject. Beginning with the first entry, Rule 4 is applied to start the NP. Rule 5 is not used, because there was no adjective supplied. Rule 6 is used, however, where an article "the" is added to the string. In applying Rules 4,6 and 8 to the second entry, an appropriate preposition must be sought. In finding a preposition, the production unit consults a table of prepositions in the lexicon. The listing contains prepositions along with a node label referring to one of the concepts shown in Figure 9.

These listings would look like the following:

```
   .
   .
   .
BEFORE (TIME)
AT (LOC TIME)
 UNTIL (TIME)
AS (EQUATIVE)
   .
   .
   .
```

Because the entry "3-MI MARK ((PP-SUBJ)(LOC))" contains a location referent, the search is made for a preposition that denotes location -- "AT." For simplicity, the search ended with the first preposition found with the correct referent. (A larger interface would not use this particular approach.) In selecting the preposition, Rule 8 could be completed, resulting in "at the 3-mile mark." Using Rule 7, we now have the "plane at the 3-mile mark" as the NP(Subj).

The use of one of the first three rules to construct a sentence from the phrases depends on the type of sentence. Rule 1 covers simple declarative and imperative sentences, with Rule 2 producing Y/N interrogatory and Rule 3 wh-interrogatory. In the case of the interrogatories, auxiliary verbs and wh- words also are obtained from the lexicon.

As with the prepositions, the wh- words are listed along with their node labels, for example:

.
.
.
WHEN (TIME)
WHERE (LOC)
WHO (AGENT)
.
.
.

The production unit is simple and limited, but it will produce sentences like the following:

DID YOU NOT SAY THE CALL?
DID YOU FORGET THE ADVISORY?
WHAT DID CAUSE THE STAMMER?
WERE YOU NOT LOOKING AT THE AZIMUTH?
DID YOU NOT REMEMBER THE PREVIOUS CALL?
WHO IS NOW SPEAKING?

In fact, it will produce a far greater variation in sentences than our simple understander can handle. Its simplicity does cause a few awkward phrasings, however, such as, "What did cause the stammer?"

One of the weak points is the selection of the preposition. If the understander passes the following:

STUDENT	: subject
LOOK	: verb
\AZIMUTH	: PP-verb
\NOT	: neg

to the production unit, along with sentence features, it produces "Were you not looking at the azimuth?" However, if the verb "WATCH" is substituted for "LOOK," (because it represents the same PROC -- SCAN), you get, "Were you not watching at the azimuth?"

Furthermore, sentences slightly more complex cannot be generated, such as:

DID YOU HAVE TROUBLE VERBALIZING THE CALL?

Something like this would be generated:

COULD YOU NOT VERBALIZE THE CALL?

Thus, as in the parser, the production unit is limited to simple sentence forms, i.e., no compound sentences, subordinate clauses, gerunds, etc.

Understander. The understander is the section of the interface that accepts instruction from the diagnostics unit and translates the instructions into a form that the production unit can use to generate a response or query for the student. It then translates the parsed input from the student into further instructions for the production of another response from the system, or sends the appropriate answer back to the diagnostics module. Figure 21 summarizes the relationship between modules.

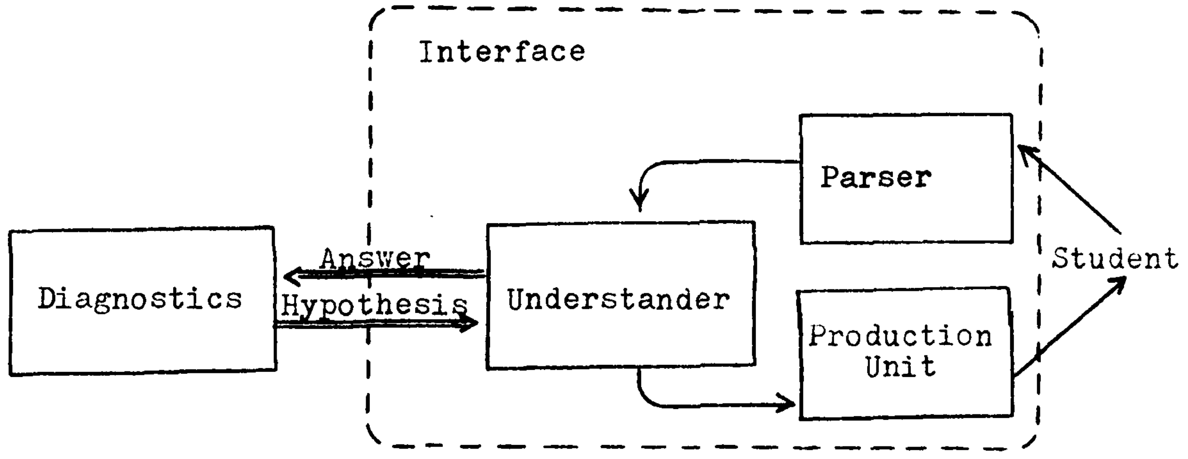


Figure 21. Overview of the relationship of the interface components with diagnostics.

The understander probably is the most limited of the three components of the interface. The parser and production unit can accept and produce much more than the understander can handle, and they easily could be expanded to do even more. It was not necessary to build an extensive deductive inference system on the current project, and so the understander was developed simplistically to deal with the types of propositions diagnostics would generate in its constant assessment of the student's status.

Recall that the task of the diagnostics component was to generate hypotheses about the causes of an error. The causes, of course, would necessarily involve one or more of the basic PROCS. The PROC that was to have been executed may have been inefficient enough to have been prematurely terminated, resulting in a missed call or a guess (using default values), which yielded an incorrect call of some form. However, the problem may have resided in a previous stage (PROC), which either passed erroneous information on to the present PROC, or inadvertently terminated the whole processing sequence before the present PROC was even activated. Additionally, the problem could lie in a subsequent PROC further down the processing sequence. Distraction, the diversion of needed resources, during the execution of an otherwise sound PROC is another possibility. To augment the function of diagnostics, an ability to interact with the student clearly was needed. This interaction naturally would concern the hypothesized problem PROC, at least initially.

The flow of execution of the principal functions of the understander can be found in Figure 22. The process would begin with the diagnostics unit passing a hypothesis to the understander and the setting of a goal which was to verify the hypothesis. The hypothesis would take the form of naming a PROC that might have been suspected as the most probable cause of the error in question. For example, the hypothesis passed might simply be DETECT.

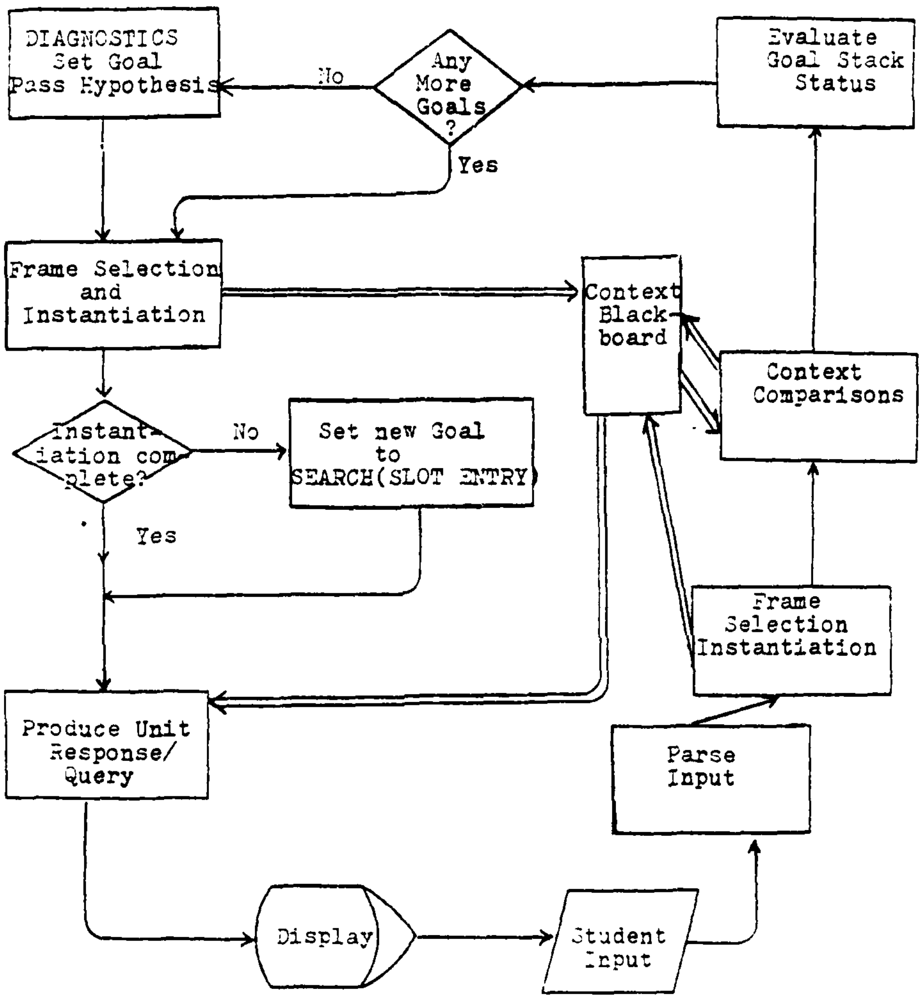


Figure 22. The major processing components of the understander.

As shown in Figure 22, the first actions taken are frame selection and instantiation. Recall that each of the PROCS are represented by frame-like data structures.

Figure 23 shows the data structure for the detection PROC as it would appear with the slots filled in except for default values.

DETECT

TYPE	?	
INPUT	SCAN	
OUTPUT	COMP	
SUBJECT	(STUDENT)	AGENT
VERB	(\PERCEIVE)	DETECT
PREP V	*12	
OBJECT	(\PLANE)	OBJECT
PREP O	(\THIS POINT)	TIME

Figure 23. DETECT frame as it is when brought in from the data base.

At this point, it would be logically equivalent to the predicate:

DETECT (TYPE, AGENT, OBJECT, RELATION, TIME).

Because the system has a specific error to which it is making reference, the argument can be given values. (It will be recalled that the diagnostics unit is giving a "playback" of the last approach to the student and has frozen the "playback" at the point that a particular error had occurred. In the present example, it was, for instance, an error in making a position call).

With the values filled in, the predicate becomes a proposition of the form:

DETECT (POS, STUDENT, PLANE, ABOVE GLIDEPATH, 3.2 MI)

which has a truth value, i.e., the proposition that the student did detect the plane as above glidepath by the time the plane was at the 3.2-mile point is either true or false. Therefore, the goal of the interrogation is to verify the proposition (determine its truth value).

Similarly, the frame when instantiated would look as shown in Figure 24.

DETECT

TYPE	POSITION	
INPUT	SCAN	
OUTPUT	COMP	
SUBJECT	(STUDENT)	AGENT
VERB	(\PERCEIVE)	DETECT
PREP V	\ABOVE GLIDEPATH	POSITION
OBJECT	(\PLANE)	OBJECT
PREP O	(\THIS POINT)	TIME

Figure 24. An example of an instantiated PROC.

In the instantiation process, the slots are filled in with the information at hand. In the example, the surface error might have been an incorrect advisory. The advisory should have been "ABOVE GLIDEPATH," and the position should have been DETECTed by mile 3.2. The initial entry for PREP V, (which was *12), represents symbolically a pointer to a program fragment that would give the understander access to the current position. The same could have been done for "time," (PREP O), but the prepositional phrase it produced, "at 3.2 mile" sounded too mechanical. Thus, the idiom "\THIS POINT," which produced "at this point," was inserted to obtain a more natural flow.

In addition to the frame selection and the instantiation processes, the instantiated frame was written to the blackboard to ultimately be used by the production unit. Production of a question would immediately ensue unless a slot entry was missing, which would cause a goal change. For the sake of continuity, however, the discussion of the handling of goal changes will be postponed until later in this section.

The production unit would take the information from the blackboard and produce a question that would be displayed for the student.

In the case of our example, the query generated was:

DID YOU PERCEIVE AT THIS POINT THE PLANE AT ABOVE GLIDEPATH?

The phrasing was somewhat awkward because of our simplistic treatment of prepositions and their connotations, and the fact that our rewrite rules in the production unit always put a prepositional phrase directly behind the word that it modified (Rules 7 and 8 in Figure 20). It seemed, however, to be a phrasing that communicated adequately. In fact, one could get used to the construction, which in turn probably could affect one's report writing skills.

Once the question is displayed, the next action taken by the interface would be the parsing of the student's input. The student could have answered with a mere "yes" or "no," which would immediately affirm or negate the proposition, thus resolving the current goal. Other possible and more complex responses are likely to occur, however.

The student could have affirmed the proposition by restating it as:

I PERCEIVED IT AS ABOVE GLIDEPATH.

In another variant, he could have used a different verb:

I DETECTED IT CORRECTLY.

He also could assume a context, as in:

I DID.

Other possible responses come from a changing of the topic in the interrogation. The student could resond with:

I DO NOT REMEMBER,

or with:

I WAS NOT LOOKING AT THE PLANE AT THE TIME.

In order to explain how the understander deals with the student input, the first two examples will be discussed first.

As shown in Figure 22, the first operations on the student's input are frame selection and instantiation. Unlike the first time, this function was called (when the frame was selected on the basis of the hypothesis), this time it is based on the main verb in the student's input. Once instantiated, it is written to the blackboard alongside (logically speaking) the previous frame that produced the query. In the first example, the student used the same verb as the system ("PERCEIVE"), thus the frame accessed would be identical. In the second example, the verb "DETECT" was used. In its search for the frame, the understander first must consult the lexicon where the verbs are referenced by the underlying PROCS that they might represent. Figure 25 shows the functional form of the verb entries in the lexicon with their associated referents.

.	
.	
.	
PERCEIVE	DETECT
DETECT	DETECT
FIGURE	MCOMP
THINK	MCOMP
PROCESS	MCOMP
REMEMBER	RETRIEVE
LOOK	SCAN
ATTEND	SCAN
.	
.	
.	

Figure 25. A portion of the verb list showing the verbs listed with the underlying PROC that they represent

It can be seen that DETECT and PERCEIVE as listed were thought to represent the same detection process. Thus, when the frame selection routine is executed, the same frame would be brought in by either of the two verbs. Using the second exemplar sentence for instantiation ("I DETECTED IT

CORRECTLY"), the blackboard would be functionally arranged as in Table 5. When written to the blackboard, the subject is changed from "I" to "STUDENT" and the verb put into present tense. Default words that come with the frame have now been replaced by the inputs from the student.

TABLE 5. STUDENT INPUT ADDED TO BLACKBOARD FOR COMPARISON

DETECT	From Hypothesis	Student's Input	
TYPE	POSITION		
INPUT	SCAN	SCAN	
OUTPUT	COMP	COMP	
SUBJECT	(STUDENT)	STUDENT	AGENT
VERB	(\PERCEIVE)	DETECT	DETECT
PREP V	(\ABOVE GLIDEPATH)		POSITION
OBJECT	(\PLANE)	IT	OBJECT
PREP O	(\THIS POINT)		TIME
		CORRECTLY	

It will be recalled that the current goal was to determine the truth value of the proposition.

DETECT (POS, STUDENT, PLANE, ABOVE GLIDEPATH, 3.2 MI).

The student has responded with a statement that can be considered a second proposition where the truth value, as specified by the student, is true. What remains is for the understander to determine whether or not the second proposition is saying the same thing as the first. To do that, it interprets the second proposition within the context of the first. In comparing the verbs, both refer to the same underlying PROC, thus both reference the same action. The agent is the student in both cases. However, the objects differ, i.e., "PLANE" does not equal "IT." The word IT is a pronoun and is designated so lexically. Thus, in comparisons, IT has no meaning in its own right and therefore acquires the meaning of the other object within the context. So the judgment is that PLANE = IT. Should the student respond with a synonym (e.g., "AIRCRAFT"), the noun list also contains a referent with each entry. Thus, AIRCRAFT would have been judged identical to PLANE. The entries for position and time are handled the same way as the object entry. The student's response did not give any position or time references. The action taken by the understander in such cases is to interpret blanks within the context as you would a pronoun. It is assumed that if the student had disagreed with either the position or time specifications, the student would have offered his own alternatives. Thus, a simple heuristic, absence of an entry, is implicit agreement.

In comparing the two propositions on the context blackboard as just described, the propositions are judged equivalent. Because the proposition given by the student was presumed to be true, the proposition generated from the hypothesis also would be true. The proposition being true then would indicate that the student did in fact perceive the plane in the position that he should have. Therefore, the original hypothesis would be false; the problem did not lie in the detection process. The fact that the hypothesis was not confirmed would be sent back to the diagnostics module, which would have the option of creating another hypothesis and starting another question-answer cycle.

The simplistic technique of comparing the student's input with the initial proposition, works well when the student responds in a way in which there is little change in the context. However, let us return to a previous example where the student's response referred to a different PROC.

The example was:

I WAS NOT LOOKING AT THE PLANE AT TIME TIME.

As can be seen in Figure 25, the verb "LOOK" represents a SCAN and not a DETECT, thus changing the context.

The sentence by itself, however, represents a proposition in its own right, which would be:

SCAN (STUDENT, PLANE, 3.2 MI)

with a truth value of "false." The student essentially has told the system the source of the problem. The understander would not be aware of that fact, however, and would simply pass the information (false proposition) on to the diagnostics component. The diagnostics component still might have the understander ask whether the student had detected the plane, if it were not for the fact that SCAN must be fulfilled before DETECTion can occur. (This information is applied in the frame as shown in Figure 23 in the "INPUT" slot.) Now diagnostics has found the source of the problem and need go no further.

The understander can handle a change in verbs and contexts as long as the student is offering another proposition about a different PROC, but if the student responds with a verb that makes no reference to a PROC, it is a little more difficult to relate it to the original question. Certain verb changes can be handled with predesigned program fragments if those changes were to happen frequently enough to be anticipated.

Such would be the case for "REMEMBER" or "KNOW" in the responses"

I CANNOT REMEMBER.

or,

I DO NOT KNOW THE REASON.

If the student were to respond with an idiom or a unique phrasing of some kind, such as:

BEATS ME!
I HAVE NO IDEA.
THE SCREEN IS NOT IN FOCUS.
I AM GETTING TIRED.

the understander then would have to respond with an indication that it doesn't understand.

It would make the same indication if the student were to respond with sentences that were too complex, such as:

I WAS HAVING TROUBLE KEEPING MY ATTENTION ON THE AZIMUTH.

In part, some of the limitations are caused by the simplicity of the understander's current design. A larger natural language interface would have no problem with that last sentence. However, it should be noted that even the most complex interfaces need to operate within a restricted domain.

To complete the description of the understander, we need to return to a point that was only mentioned briefly. Remember that in the course of filling frame slots, there was the possibility that there was insufficient information available with which to satisfy the slot requirements. When such an instance occurred, Figure 22 indicated that a new goal was to be formulated. The interface possessed a push-down goal stack by which it kept track of its current intentions. When a hypothesis was passed to the interface from the diagnostics module, the goal placed in the goal stack was to verify the proposition of the PROC which was named (denoted VERIFY(PROC)). Before the goal could be resolved, another need may arise such as the need to get information from the student regarding a slot entry. In that case a new goal (e.g., SEARCH (SLOT ENTRY)) would be placed on the top of the stack, pushing the old one down.

With the goal set for the search for a slot entry, the production unit uses the information to construct a wh-question like

WHAT DID YOU PERCEIVE?

with the exact form (what, where, etc.) being dependent on whether the missing slot entry was a subject, object, time, etc. Once the goal on the top of the stack was resolved, it was removed, allowing the old one to emerge again. The dialogue between the interface and the student then would continue until all the goals in the stack had been removed, or a comment from the student (such as, the student's not remembering anything) caused the process to exit. The types of goals the interface was designed to handle were: VERIFY, TELL, SEARCH and COMMAND, with varying arguments.

Adjusting the Student Model. Up to this point we have focused on the diagnosis of the causes of errors. However, the CRAI's adjustment of its model of the state of the student's cognitive processes is not solely error-driven. It is the nature of the task environment that errors are indicative of discrepancies between what the CRAI determined the student should be capable of, and the student's actual abilities, i.e., a discrepancy between CRAI's student model and the actual state of the student. Furthermore, because any one of the hypothesized cognitive processes could cause a task failure, it is the nature of the inference process that numerous and complicated algorithms are necessary to discern the nature of this discrepancy. Therefore, a large portion of a trainer development effort can be expected to be devoted to developing these algorithms. On the other hand, pedagogically, the students successful task performance is strategically important. Indeed, if the student exceeds expectations this also indicates a discrepancy between the model and the student's actual state. However, because all of the cognitive components must function satisfactorily to yield a successful performance, the inference process is vastly simplified: one can assume an increment in the proficiency of each contributing component.

The psychological literature, reviewed in our discussion of the essential student features in Section II, suggests that in cognitive processing, proficiency should be positively correlated with processing speed. Indeed, reaction time or response latency has been used in many studies as a dependent measure of proficiency. The present task environment precludes the use of a precise response latency measure because the target stimuli are continuous, i.e., there is no onset point from which to measure latency. One approximation, of this type of measure is available by measuring the time between successive advisories. If an advisory is missed no measurement can be made for that "non-advisory" or for the next advisory. In this case, some default estimate is needed. We shall call the measurement associated with an observed event (or non-event) the proficiency measure (PM) for that event.

The assumption of an action sequence leading to the observed response, implies that the PM is a function of the proficiencies of the components of that sequence. Similarly, the expected proficiency (EP) measurement for a projected response would be a function of the expected proficiencies of each of the components of the sequence leading to that response. One of the attributes of each of the PROCS in the IKB is a proficiency score. The initial value of this score is an estimate of a cognitive component's contribution to the EP of a given action sequence. EP is the sum of the proficiency scores of each of the components of the action sequence.

Figure 26 illustrates the four logical outcome possiblities and the associated implications for adjustment of the student model. The first step in the adjustment procedure is to build a model of the particular action sequence leading to the observed response. If a response is missing then the model assumed is that of the response which would have been appropriate. As outlined in our discussion concerning elaborating the student model, one of the attributes of each PROC in the IKB is INPUT. The value of this attribute is a pointer to the specific PROC which led to the present one. To build a model of the response's action sequence, the CRAI starts with the observed, or appropriate, verbal response. It then traces back via the INPUT attribute the sequence of PROCS that preceded it. In parallel with this search, the CRAI retrieves and sums the proficiency score of each component PROC. The resulting sum is the EP associated with verbal response.

	PM < EP	PM > EP
ERROR DETECTED	No Adjustment is Made	Adjust the Cognitive Component(s) of Cause
CORRECT ADVISORY	Adjust All Components	No Adjustment is Made

Figure 26. Possible outcome combinations.

The default estimate, if needed, is a function of the EP's for all of the advisories required on the approach. The rationale of this default is a product of the method of computing the task scenario which will be explained later. If a single PROC is determined to be a source of error, then the entire difference between the PM and EP is assumed to be associated with this PROC. The difference is then added to the proficiency score of the PROC. If more than one PROC remain candidates for the cause after analysis and interrogation, then the PM-EP difference is allocated between the PROCS in proportion to their relative contributions to the EP. The assumption is that variance in a PROC's proficiency score would be proportional to that score. Similarly, if a verbal response is successful and the PM is less than the EP the proficiency scores of the component PROCS are proportionately decremented by the absolute difference. No adjustment is made when an error is accompanied by a PM which is less than the EP. The error inherently indicates that an improvement in the EP is not called for. Subsequently, an appropriate decrement for the EP can not be determined in this case. No adjustment is made when a successful advisory is accompanied by a PM which is greater than the EP. This situation may arise because the paced nature of the task dictates a PM which is worse than the student's actual capabilities would warrant.

THE CURRICULUM DRIVER. The curriculum driver utilizes the adjusted student model to direct a search through the possible approach scenarios in order to select the one which should yield the greatest improvement in student learning. It will be recalled that the results of our simulation experiment with the turnpike solution indicated that the greatest increment is achieved on each trial by applying the greatest resources to the least learned component. The curriculum driver applies this principle to scenario generation.

The various approach scenarios can be represented by a tree diagram (see Figure 27). The <u>nodes</u> of the tree represent the possible values of the denoted task parameters. Figure 27 is only a partial tree for illustrative purposes. The actual tree, representative of the INSTRUCT task would have seven elevation nodes, each with subsequent branches. Note that the number of possible branches grows geometrically at successively lower levels of the tree.

The curriculum driver is representative of a branch and bound search technique. The driver searches the branches of the scenario tree subject to the constraints of optimizing the expected learning of the student during the next practice task and the physical realities of the task. A primary goal of any search technique is to find the optimal solution to a problem as efficiently as possible. In a tree search this essentially means exploring as few branches as possible without missing that branch which leads to the optimal solution. Because the number of branches increases geometrically as one moves from the start node toward the myriad termination nodes, this efficiency is achieved by ordering constraining contingencies such that they limit the possible branches to be explored as early as is feasible. Feasibility is a function of the probability of overlooking the optimum solution and the value of finding that solution.

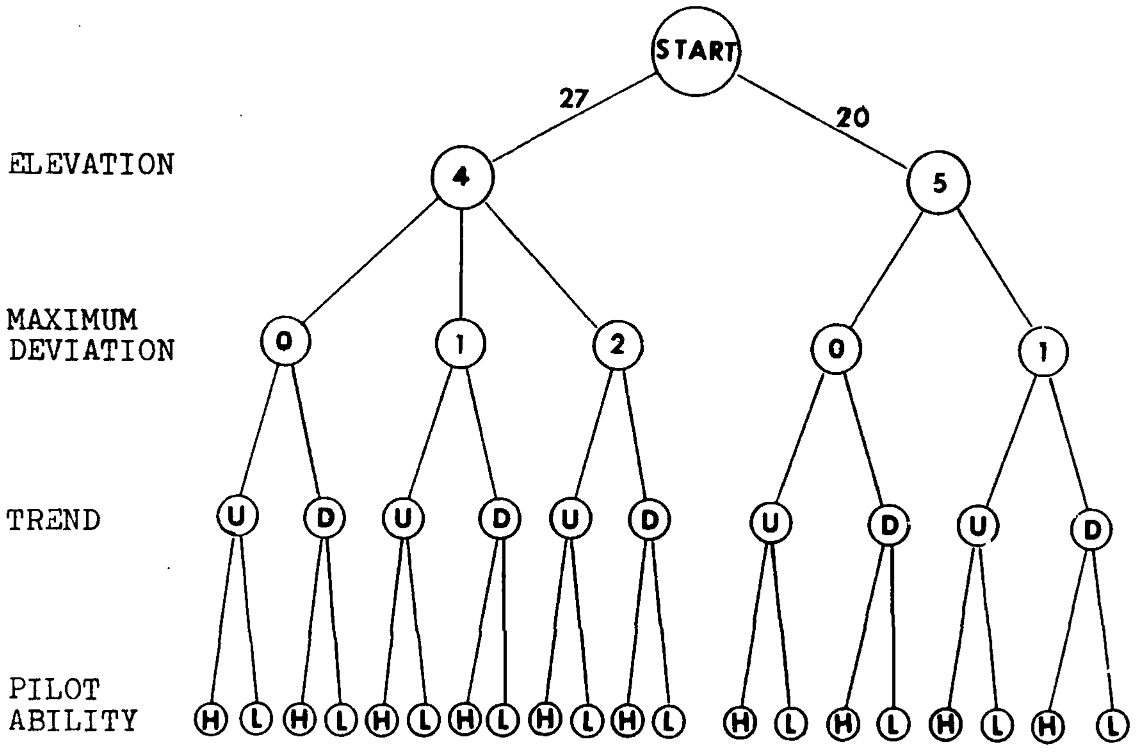


Figure 27. A scenario tree.

In INSTRUCT the constraints of the physical realities of the task are constant, but optimizing the expected learning of the student is contingent upon the current learning state of that student. This state is represented in the student model.

The first step then is to have the curriculum driver retrieve the EP for each of the seven elevation advisories. This is done in the same way as explained for model adjustment. For each combination of task parameters, i.e., each approach scenario, a pattern of ten verbal responses with their associated EP's can be generated. The sum of these ten EP's for a given approach scenario would be indicative of the overall learning level associated with that scenario. Note that since these values are representative of a latency type measure they would be negatively correlated to proficiency, i.e., a higher number indicates a lower proficiency.

As per our earlier description of the task, the flightpath of the target plane varies around a selected average glidepath. A trivial optimal solution to the search would be to select the elevation with the highest EP and maintain the target at that level for the entire approach. However, it would seem psycnologically reasonable to assume that the second successive advisory for a given elevation would not require the same degree of processing as the first, since various encoding steps would not have to be repeated. This is certainly true for the simulated student. Therefore, the curriculum driver reduces the EP for a successive advisory for the same elevation. The trivial solution is a logical boundary. It illustrates that any other optimum solution would have to be constructed around the elevation with the highest EP. Thus, the first search constraint is to select the branch corresponding to this elevation. The values indicated next to each of the top branches in Figure 27 represent the EP associated with the indicated elevation. The constraint eliminates the necessity for searching the six remaining branches and their many associated scenarios.

Once the first branching decision is made, the only remaining constraints which may be exploited are the physical constraints of the task. For example, in our task the target is not allowed to leave the glidepath completely. Therefore, a scenario which would call for an average elevation of above glidepath with a maximum deviation of two zones above and below this elevation would not be permitted. Thus it is possible to constrain the search from examining all possible combinations of task parameter values. These constraints may be efficiently ordered. The selection of the average glidepath and a maximum deviation constrain whether the initial trend will be rising or falling. Setting these three parameters constrain whether the plane can cross the average glidepath. In the above example, a maximum deviation of two zones would be acceptable if the trend was downward, and if the target never rose to the maximum deviation above the glidepath.

Within the constraints outlined above, an EP sum is calculated for each feasible scenario. The curriculum driver then simply selects the scenario with the largest associated EP sum.

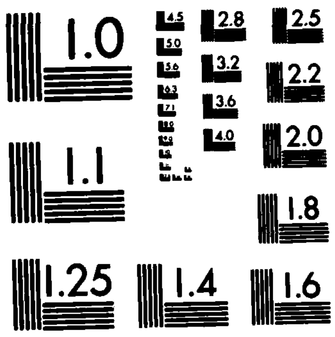
AD-A124 126 INSTRUCT: AN EXAMPLE OF THE ROLE OF ARTIFICIAL INTELLIGENCE IN VOICE-BASE..(U) BEHAVIORAL EVALUATION AND TRAINING SYSTEMS LUBBOCK TX D C CHATFIELD ET AL. 2/2

UNCLASSIFIED JAN 83 NAVTRAEQUIPC-80-C-0061 F/G 5/9 NL

END

FILMED

DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

The four task parameters, average glidepath elevation, maximum glidepath variation, initial glidepath trend, and pilot ability, are representative of what we call <u>state determined</u> task variables. Their selection is a function of the student's learning state, as that state is represented in the student model. However, an optimal pedagogical strategy must be based not only on the present state of the student but on the rate at which that state will change. Two students may start at the same state but their rate of progress may differ. The CRAI should be able to adapt the task to each individual so as to progress that individual as quickly as the individual's abilities allow. In the INSTRUCT task the plane's speed is such an <u>adaptive</u> parameter. Once the state parameters are determined, a too slow speed will unnecessarily impede the student's progress toward full task speed ability. On the other hand, a too fast speed will prevent the student from ever fully encoding advisories. Hence, it too will impede the student's progress. One test of the CRAI's pedagogical effectiveness is its ability to detect differences between students and adapt its scenario generation accordingly.

SECTION V

SIMULATIONS AND RESULTS

STUDENT BEHAVIOR

Our first experiment was to examine if the simulated student exhibited behaviors representative of novices. Therefore, we sat the student down at the display tube and observed its behavior. Before its first approach, the simulated student has read the instructional manual, i.e., it "knows" all of the appropriate advisories and the rules governing when they are to be issued. However, it has had no practice doing any of the cognitive acts required by the task. Its process learning level is zero.

Table 6 contains a response protocol taken from the third approach of the student. The second column contains the exact glidepath elevation of the target. For example a target with an elevation which is greater than or equal to 3 but less than 4 is in zone 3.

TABLE 6. A REPRESENTATIVE VERBAL REPONSE PROTOCOL.

DISTANCE	POSITION	ADVISORY
9.98162	5.99815	MM...ABOVE GLIDEPATH
8.93772	4.32417	UH...SLIGHTLY ABOVE GLIDEPATH
7.93302	3.01148	WELL BELOW GLIDEPATH
6.84185	4.89402	SLIGHTLY ABOVE GLIDEPATH
5.926	5.98683	UH...WELL BELOW GLIDEPATH
4.99747	4.47103	ABOVE GLIDEPATH
.988574	4.45564	SLIGHTLY ABOVE GLIDEPATH

Three instances of phrasing problems are immediately apparent in this protocol on advisories one, two, and five. Admittedly, real students' phrasing problems do not look like these. However, as far as the CRAI is concerned these representations are just as uninterpretable as any more elaborate production would be. Therefore, no attempt was made to emulate more elaborate behavior. Two forms of phrasing errors can be seen. The "MM..." form indicates to the experimenter that the student had completed verbal encoding before the response was made. The "UH..." form indicates encoding was not complete. Notice that the second advisory is correct, other than phrasing, even though the form definitively indicates it was a guess. The distinction between forms was solely to aid our analysis of CRAI performance. The CRAI can not distinguish between the two forms. We have assumed a worst case voice recognition system with limited interpretive flexibility, for testing our implementation ideas. These verbal production problems completely disappear with training.

Next, one will notice that the student tends to make all of the advisories slightly late. The slight time lags represented here were defined as acceptable. An unacceptable lag would be equal to or greater than half of a mile. With training, not only does the student stop making advisories late (and stops missing them) but the time lag is significantly shortened.

Although there were no definitionally late advisories in this protocol, three advisories were missed. They should have occurred at markers four, three and two. We found that the simulated student gets behind as real students do. This behavior also often is preceded by an error, which forecasts a problem exists.

The student made two outright errors in this protocol on advisories three and six. The first error is obviously a guess. A guess this wild may not be frequent among real students. We did not program the CRAI to distinguish between close guesses and wild ones. Thus, this type of possibly unrepresentative behavior was not deemed fatal in any way. The second wrong advisory could be caused by a number of factors. Its phrasing is correct, but it still could be a guess. A second possibility is that the target was inaccurately detected. These two events simulate the observation of Chatfield et al. (1979), that wrong advisories could occur from processing errors even with perfect knowledge (albeit not necessarily perfect knowledge organization). As would be expected, wrong advisories cease with practice.

The student has also managed to make two completely correct advisories on advisory four and ten, even though an error is made on this same elevation on advisory six. Correct advisories, of course, increased with practice.

CRAI INFERENCE.

We examined the CRAI's ability to infer the covert causes of the student's mistakes. We tested this ability under the worst case condition, i.e., that the student cannot introspect at all. We did this for two reasons. First, it is the most severe test of the CRAI's inherent inferential ability. Second, any degree of introspection introduced would be arbitrary and would limit interpretation. We also imposed a second difficulty on the CRAI, by examining the inferential ability during early student approaches. It is during these approaches that phrasing/speech degradation is the greatest problem. Without the interrogatory interface the CRAI can not distinguish an ill-phrased response from a speech-degraded response. However, phrasing/speech degradation was included as a diagnostic event in the CEDS. We wished to see if the system could inherently extract meaningful information from these advisories. Four approaches were examined: two approaches each from two students of different learning abilities.

The first analysis is summarized in Table 7. The CRAI often proposed more than one possible cause. However, an examination of the student's cognitive monitor revealed that some errors in fact had more than one cause. A "hit" was defined as any time the CRAI indicated a true cause. A "miss" was defined as any time the CRAI failed to infer a true cause. We did not analyse the number of "false alarms", i.e., the number of times the CRAI indicated a false cause. If introspection were allowed, most likely a large portion of these would be dismissed. However, the size of that proportion can not be unambiguously determined since it would depend on the students' ability to introspect.

As can be seen from the table, the CRAI did quite well in this analysis. Overall it correctly identified the cause of an error 75 percent of the time. Phrasing problems did tend to inhibit the inferential ability to some degree. The hit rate decreases to 72.4 percent under this condition. This would be expected. Because the system can not inherently interpret the semantic meaning of an ill-phrased verbal response, correct responses can not be distinguished from incorrect ones. As mentioned in our description of the diagnostic module, the interrogatory interface was developed to resolve this type of ambiguity. Through questioning the student, the CRAI could determine the meaning of the ill-phrased response. Indeed, there is a diagnostic value to the distinction between an ill-phrased but otherwise correct response and an incorrect one. We would expect this additional information would raise the hit rate on advisories with phrasing problems.

TABLE 7. TABULATION OF HITS AND MISSES BY PHRASING.

	CAUSES HIT	CAUSES MISSED	
ILL-PHRASE/DEGRADED	21	8	
CLEARLY PHRASED.	9	2*	
	30	10	40

*One advisory was a correct guess

When the verbal response was not obscured by phrasing problems, the CRAI scored a hit 81.8 percent of the time. One of the two misses, in fact, would have probably slipped by a human instructor, since the verbal response was correct. Only by examining the cognitive monitor was it possible to determine this response was really a guess.

If the CRAI simply hypothesized all causes all the time it would never miss. Table 8 lists the eight different hypotheses available to the CRAI. The last six of these hypotheses are indicative of covert causes.

TABLE 8. POSSIBLE CRAI HYPOTHESES.

OVERT CAUSES	COVERT CAUSES
Late Previous Call	Inefficient DETECT
Late Boundary Crossing	Inefficient PCOMP
	Inefficient CALL
	Didn't SCAN area
	Late SCAN
	Incorrect Structure.

Table 9 reveals that the CRAI never made more than three hypotheses. In fact, 22.5 percent of the inferences were single hypotheses. The CRAI scored a hit on five out of the seven. However, because of multiple-cause errors, it missed four true causes.

TABLE 9. TABULATION OF THE NUMBER OF HYPOTHESES PER ERROR BY TYPE OF PHRASING.

HYPOTHESES	ILL-PHRASED/ DEGRADED	CLEARLY PHRASED	TOTAL
SINGLE	6	1	7
DOUBLE	13	1	14
TRIPLE	1	9	10

We also examined the effect of phrasing difficulties on the number of hypotheses issued for each error. Table 9 also summarizes this data. It appears that phrasing/speech degradation can have diagnostic value, at least regarding our simulated student. This is indicated by the higher frequency of single and double hypothesis inferences when phrasing/speech degradation problems are identified.

SCENARIO ADAPTATION

Next we examined the CRAI's ability to distinguish between students of different abilities and to adapt the training curriculum accordingly. Figure 28 illustrates the relative verbal encoding learning curves for the three students.

```
100        +++   ****      -------
96        +    **       ---
92       +   **      ---
88          *      --
84      +  *     --
80        *     -
76     +      --
72       *   -
68    + *   -
64         -
60     *  -
56
52   +   -
48    * -                   + Top student
44                          * Mid student
40     -                    - Low student
36   *
32  + -
28
24  *-
20
16  -
12
8
4
0
```

Figure 28. Verbal ncoding progress rate of three simulated students.

We first examined the CRAI's decisions regarding the state parameters. We found no change in the pilot ability factor either within or between students. This factor affects the degree to which the target crosses the average glidepath elevation. The CRAI kept the pilot ability low (i.e., the target crossed often). This merely indicates that the students' learning level for advisories above and below the average glidepath were roughly symmetrical. The initial trend parameter showed no meaningful pattern. This would be expected in view of the pilot ability factor. In addition this factor serves largely as a control factor constraining the target when the average glidepath elevation is near the extremes.

The state parameters whose patterns are most revealing are those for average glidepath elevation and maximum deviation. These results are summarized in Table 10. A clear trend is apparent. First, all students start with the same scenario. This is expected because the learning state for all three students was initially identical, only their learning rates differed. The CRAI appears to have distinguished this fact. Note that the top student is advanced to covering below glidepath advisories immediately following the first approach. The middle student isn't advanced until the fourth approach, and the lowest student not until the fifth. Also of note is the way in which the CRAI manipulated the deviation factor in conjunction with elevation. The top student is given broad experience up to the third approach, when the CRAI apparently determined the necessity for more specific practice. For the lower student this narrowing of task focus comes earlier, and the focus is on repeating the above glidepath advisories. The CRAI appears to be saying, "Keep going over these advisories until you get them right."

TABLE 10. STATE PARAMETERS FOR THREE DIFFERENT STUDENTS.

TOP STUDENT	MID STUDENT	LOW STUDENT
4/2*	4/2	4/2
2/2	4/1	4/2
2/2	5/1	4/1
1/1	2/1	5/1
5/1	1/1	1/1

*The first figure is the average glidepath elevation. The second figure is the maximum zone variation.

The CRAI's adaptive ability is further evidenced by its manipulation of the adaptive parameter -- speed. These results are summarized in Table 11. For the top student the CRAI increased the speed of the plane almost constantly. The one place where speed was decreased was when the CRAI had the student repeat an approach scenario. For the middle student, the CRAI kept the speed fairly constant. Recall, that the CRAI tends to iterate towards a representative model of the student. Thus the CRAI will tend to test and adapt, test and adapt etc, precluding a perfectly constant setting. For the lowest student it is clear the CRAI initially sensitively dropped the speed parameter below that of the other two students. The setting is never raised to that of the top student, but on the last two appoaches it is above that of the middle student. We don't have a good explanation of this result at this time. Considering the reasonable behavior of the CRAI outside of these two events, further examination is warranted to determine if they are artifacts of some atypical circumstances.

TABLE 11. PLANE SPEED ADAPTATION FOR THREE STUDENTS.

TOP STUDENT	MID STUDENT	LOW STUDENT
1.22	1.22	1.22
1.24	1.18	1.02
1.21	1.21	1.15
1.43	1.18	1.32
1.45	1.21	1.37

SUMMARY

We have tested INSTRUCT on a number of critical dimensions. First, the results indicate that we have developed a simulated student from a small number of primitive processes, the behavior of which appears to be remarkably representative of real students. Second, through the implementation of a number of basic AI techniques we have been able to develop a CRAI with the ability to reasonably determine possible covert causes of errors from overt responses. In addition, the work suggests that some useful diagnostic information may be available even from responses the CRAI can not semantically identify. Fourth, the CRAI appears to be able to translate its sensitivity to the students' behavior into reasonable pedagogical strategies.

SECTION VI

CONCLUDING REMARKS

Our speculations, ideas and conclusions have taken us over a wide array of topics and technologies. In attempting to develop a simulation of a small voice-based trainer, we were confronted with numerous practical problems that would send us back to the drawing board from time to time. We hoped that through the process of designing an intelligent model of the instructor for voice-based trainers (albeit a small simulator), certain ideas and technologies could be identified. By trying specific approaches and making recommendations, we have attempted to expedite any subsequent effort to develop and implement an intelligent instructor model for an automated trainer based on voice recognition. For that reason, we would like to conclude with a discussion of our ideas and recommendations regarding further research, development and enhancements in technology that would be required in any fully implemented trainer. First, however, is a review of the ideas that developed during our current effort.

Our experience with INSTRUCT led us to delineate five functional areas of the model of an instructor as described in Section IV. Those functional areas or modules are: the data-base, the student model, the curriculum driver, the diagnostics module, and the natural language front-end. Within each of these areas, we reviewed the functional needs, some of the existing technology that could be utilized and our particular set of ideas or approaches.

Our conceptualization of the specific training problems came from the interviews, observations and literature reviews reported mainly in Chatfield, et. al. (1979). These findings led to certain assumptions about the nature of the types of tasks and requisite information-processing skills to which voice-based trainers typically might be applied. These assumptions are discussed in some detail previously, but will be reviewed here briefly:

1. The tasks, to a great extent, are event-driven, requiring highly practiced cognitive, verbal and, in some cases, motor skills.

2. Each response that is made (e.g., an advisory) is assumed to be the product of a series of decomposable processes, some of which are identified in earlier sections.

3. Each of these processes is assumed to require both time and resources for execution.

4. The amount of resources and time required for a process diminishes with practice approaching automaticity.

5. The rate at which the requirements diminish is positively related to the amount of resources invested during practice.

6. Resource allocations can be controlled to some extent by the student and manipulated by the system.

From these assumptions, it was reasoned that an error response might be the result of a deficiency in one or more of the PROCS in the sequence. However, the error can be caused by a sudden reallocation of resources to a concurrent sequence containing one or more deficient PROCS. Finally, if a training system were to have the *diagnostics* capability of identifying deficiencies in individual PROCS, then that information can be utilized during the generation of subsequent scenarios.

Our resulting conceptualization of the task led to the proposed prominence of the diagnostics function. An early approach to the diagnostics problem regarding processing skills was to look for error patterns that would reflect causes. This would have allowed us to identify adaptive production systems with learning capabilities, which would accumulate additional error patterns, as one of the AI technologies. We found, however, that the errors could not be traced backward to a unique identification of causes with any degree reliability. The system would generate a probability distribution over the range of possible causes, but at times the probability estimates for various causes were quite close. For this reason, we have identified the technology behind natural language as one that will be extremely fruitful for diagnostics purposes, as well as other uses. The diagnostic ability of the system to interrogate the student was augmented by the use of inter-call latency measures that proved useful in the scenario-generator and adaptive logic. These latency measures are useful in a PARTS type of task where the student is required to respond at a particular rate. This may not be feasible on other types of tasks. However, the other training situations may enjoy different opportunities for diagnostics and adaptive control by being amenable to the secondary task methodology or possibly the use of reaction time measures where the onset of an eliciting stimulus could be identified.

Any recommendations concerning further efforts will have to include natural language and its use in diagnostics as well as other functions. Because of its promise, we believe that the student's capabilities for introspection and prognostication should be determined empirically. The system's ability to talk with the student, as a human instructor would, opens up a new resource of information that training systems from a wide spectrum of applications could use.

SUGGESTIONS FOR FURTHER RESEARCH AND DEVELOPMENT

In short, we have advocated that the design of voice-based training systems be quite different from past adaptive trainers, CAI-based systems, or even ICAI-based systems. We have proposed that the training systems for these event-driven, cognitively-complex tasks, be designed to intelligently manage the student's resources during training via verbal instructions, adaptive adjustments, and appropriate scenario generation.

We have demonstrated that state-of-the-art psychological theory and AI techniques are sufficiently developed to allow us to implement a small simulation package, i.e., INSTRUCT, which successfully incorporates our theoretical ideas concerning the model of a computer resident automated instructor in a voice based trainer. We believe that continued efforts toward the implementation of a CRAI, along the lines outlined in this report, would be highly beneficial to Navy training programs. Below, we have set forth some ideas on areas and directions for future development efforts.

Future development efforts could take two directions. First, if a specific training environment is identified as needing automatization, the specific development effort could be used as a testbed for additional CRAI development.

We believe INSTRUCT has demonstrated that we, at least, currently can begin the implementation process. We anticipate that task-specific problems will arise (in the paragraphs below we outline some general areas for future development that we anticipate will encompass these problems). However, we also anticipate that these problems will not be overwhelming. In addition, we believe that the solutions achieved in a specific development effort will be generalizable and facilitate CRAI implementation in additional training environments.

A second direction would be a general development and testing effort, should it be determined that a candidate training environment should not be singled out for experiments in implementation. Below we suggest certain general areas for additional development that will facilitate implementation in future specific environments.

We suggest four general areas for further development. First, is the area of managing student resources during the training of cognitive skills thereby minimizing training time (cost) while maximizing terminal proficiency.

Before any production system could be implemented in a curriculum driver, a contracting firm would need to work with the subject matter experts (SMEs) in creating an inventory of: terminal abilities, entering abilities, knowledge and skill units, scenario contexts (and their attributes), adaptive variables (and the extent of their potential adjustment), performance indices, skill interrelationships (are they independent or competing?), vocabulary, and human instructors' pedagogical heuristics.

Given this information, a production system would need to be designed which would allow the intelligence of the system to explore instructional strategies, beginning with those known to be effective in training cognitive skills. We reviewed some of these ideas in Section II and in Chatfield et al. (1979).

The basic cognition literature also contains suggestions for training strategies. A good recent source is the collection of papers presented at the Sixteenth Annual Carnegie Symposium on Cognition which as edited by Anderson (1981).

The human instructors working in the candidate training areas are an additional source of ideas for initial training strategies which come from their subjective experiences as an instructor.

Implementation would require that our curriculum driver be able to start with training strategies that are at least reasonably effective for training the skills in question. It would also be desirable for the curriculum driver to have the ability to gradually develop more efficient strategies through the knowledge it gains by interacting with students.

The logic in INSTRUCT is capable of training various simulations of the student. However, an empirical test, with real students, has not been attempted. This could be done during implementation and development of a future trainer, or tests could be done through modifying an existing trainer such as the Ground Controlled Approach Controller Trainer System (GCA-CTS). An experimental laboratory task which embodies the essential cognitive demands found in most voice-based trainers could also be created. Any of these options would give the opportunity for testing, and further development of the requirements for a curriculum driver in an intelligent CRAI as outlined in this report.

A second area for further tests and development would be that of the diagnostics function and the attending natural language interface. As was described in Section IV, INSTRUCT possessed a very limited inference system in diagnostics and utilized a very crude interrogatory interface. INSTRUCT still was capable of reasonably accurate diagnostics. Implementation would require that a more fully developed inference engine and natural language interface be adapted for diagnostics use. While this adaptation could be done during a specific implementation, much could be done under general development. This effort would involve the definition of the domain of discourse and inference. The adaptation could be done by using existing training domains (such as PARTS) as base line references for an expanded hypothetical task. This would be an elaboration of the method used in INSTRUCT. Once the interface and inference engine were adapted, it would require little additional effort to modify the data base and make the minor adjustments required for implementation in a particular training environment.

The third recommendation is to explore the student's introspective capabilities. This can be done by selecting an actual task, similar to that used in the simulation of INSTRUCT, which would have the capability for utilizing diagnostic information in the adaptive functions, scenario generation and even misconceptions in cognitive structure. A human instructor could be present to provide the ability to carry on diagnostic discourse with the student. The amount of discourse could be varied in amounts from zero to extensive, and in nature. Unlike our simulated student, the accuracy of diagnostics could not be determined directly. Thus, indirect measures, such as performance measures reflecting learning rates as a result of the use of the diagnostics, would have to be used. Students would be expected to provide very useful information during training, as most anecdotal testimony will corroborate. More importantly, however, the effort could explore the concept of student-system discourse in discovering the relevant techniques and pitfalls of interrogation. A recent review on the issues and data concerning the use of verbal reports, interrogation, and introspection is given by Ericsson and Simon (1980).

A fourth recommendation concerns the creation of functional specifications for the type of information needed and the organization required in the implementation of an intelligent trainer. Before implementation and creation of any trainer, contractors must investigate the subject matter and its setting to be able to assemble the needed information. It would be beneficial if they had a set of design guides to give direction. When analyzing the tasks that the students are to master, it would be helpful if they could refer to the basic processing primitives and how they relate. Thus, it would be advisable to develop design guides for the representation of basic cognitive processes and structure for use in an intelligent system. This would require a broader and more systematic approach to extend the initial concepts and ideas identified in INSTRUCT. The basic research literature and theory, such as Norman and Shallice (1980), needs to be incorporated into an applied model of process representation.

REFERENCES

Allen, J. Anatomy of LISP. New York: McGraw-Hill, 1978.

Anderson J.R. (Ed.) Cognitive Skills and Their Acquisition, New Jersey: L. Erlbaum Assoc., 1981.

Anderson, J.R., & Bower, G.H. Human Associative Memory. Washington, D.C.: V.H. Winston & Sons, 1973.

Briggs, G.E., & Johnsen, A.M. On the nature of central processing in choice reactions. Memory and Cognition, 1973, 1, pp. 91-100.

Broadbent, D.E. Perception and Communication. New York: Pergamon, 1958.

Bruner, J.S., Goodnow, J.J., & Austin, G.A. A Study of Thinking. New York: Wiley, 1956.

Burton, R.R., & Brown, J.S. Natural-language capability for computer-assisted instruction. In H.F. O'Neil, Jr. (Ed.) Procedures for Instructional Systems Development. New York: Academic Press, 1979.

Carbonell, J.R. AI in CAI: An artificial intelligence approach to computer-aided instruction. IEEE Transactions on Man-Machine Systems, 1970, MMS-11, pp. 190-202.

Chant, V.G., & Atkinson, R.C. Optimal allocation of instructional effort to interrelated learning strands. Journal of Mathematical Psychology, 1973, 10, pp. 1-25.

Chatfield, D.C., & Gidcumb, C.F. Optimization Techniques for Automated Adaptive Training Systems. Technical Report NAVTRAEQUIPCEN 77-M-0575, 1977.

Chatfield, D.C., Marshall, P.H., & Gidcumb, C.F., Instructor Model Characteristics for Automated Speech Technology (IMCAST). Technical Report, NAVTRAEQUIPCEN 79-C-0085-1, 1979.

Chomsky, N. Three models of the description of language. Proceedings of a Symposium on Information Theory. IRE Transactions on Information Theory. September, 1956, IT-2(3), pp. 113-124.

Collins, A.M., & Quillian, M.R. Retrieval time from semantic memory. Journal of Verbal Learning and Verbal Behavior, 1969, 8, pp. 240-247.

Craik, F.I.M., & Lockhart, R.S. Levels of processing: A framework for memory research. Journal of Verbal Learning and Verbal Behavior, 1972, 11, pp. 671-684.

Ericsson, K.A., & Simon, H.A. Verbal reports as data. Psychological Review, 1980, 87, pp. 215-251.

Fahlman, S. Computing facilities for AI: A survey of present and near future options. AI Magazine, 2, pp. 16-23, 1981.

Gopher, D., & North, R.A. Manipulating the conditions of training in time-sharing performance. Human Factors, 1977, 19, pp. 349-365.

Greeno, J.G. The structure of memory and the process of solving problems. In R.L. Solso (Ed.) Contemporary Issues in Cognitive Psychology. Washington, D.C.: V.H. Winston & Sons, 1973.

Hooks, J.T., Butler, E.A., Gullen, R.K., & Petersen, R.J. Design Study for an Auto-Adaptive Landing Signal Officer (LSO) Training System. Technical Report, NAVTRAEQUIPCEN 77-C-0109-1, 1978.

Hunt, E. What kind of computer is man? Cognitive Psychology, 1971, 2, pp. 57-98.

Kahneman, D. Attention and Effort. Englewood Cliff, New Jersey: Prentice-Hall, 1973.

Kintsch, W. The Representation of Meaning in Memory. Hillsdale, New Jersey: L. Erlbaum, 1974.

LaBerge, D.L. Attention and the measurement of perceptual learning. Memory and Cognition, 1973, 1, pp. 268-276.

Lakoff, G. Irregularity and Syntax. New York: Holt, Rinehart & Winston, 1970.

McClelland, J.L. On the time relations of mental processes: An examination of systems of processes in cascade. Psychological Review, 1979, 86, pp. 287-330.

Meyer, D.E. On the representation and retrieval of stored semantic information. Cognitive Psychology, 1970, 1, pp. 242-300.

Miller, G.A. The magical number seven. Psychological Review, 1956, 63, pg. 81.

Minsky, M. A framework for representing knowledge. In P. H. Winston (Ed.) The Psychology of Computer Vision. New York: McGraw-Hill, 1975.

Mitchell, T., Carbonell, J., & Michalski, R. Special Section on Machine Learning. ACM SIGART Newsletter, 1981, 76, pp. 25-54.

Munro, A. Linguistic theory and the LNR structural representation. In D.A. Norman, & D.E. Rumelhart (Eds.), Explorations in Cognition. San Francisco: W.H. Freeman & Co., 1975.

Navon, D., & Gopher, D. Interpretations of Task Difficulty in Terms of Resources: Efficiency, Load, Demand, and Cost Composition. Technical Report, AFOSR-78-1 Technion, Israel Institute of Technology, 1978.

Neisser, U. Cognition and Reality. San Francisco: W.H. Freeman & Co., 1976.

Neves, D.M., & Anderson, J.R. Knowledge compilation: Mechanisms for the automatization of cognitive skills. In J.R. Anderson (Ed.) Cognitive Skills and Their Acquisition. New Jersey: Lawrence Erlbaum Assoc., 1981.

Newell, A., & Simon, H.A. Human Problem Solving. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1972.

Nilsson, N.J. Principles of Artificial Intelligence. Palo Alto Calif.: Tioga Publishing Co., 1980.

Norman D.A. Categorization of action slips. Psychological Review, 1981, 88, pp. 1-15.

Norman, D.A. Notes toward a theory of complex learning. In A.M. Lesgold, J.W. Pellegrino, S.D. Fokkema, and R. Glaser (Eds.) Cognitive Psychology and Instruction. New York: Plenum Press, 1978.

Norman, D.A. (Ed.) Perspectives on Cognitive Science. Norwood, N.J.: Ablex Publishing Corp., 1981.

Norman, D.A., and Bobrow, D.G. On data-limited and resource-limited processes. Cognitive Psychology, 1975, 7, pp. 44-64.

Norman, D.A., Rumelhart, D.E., & The LNR Research Group, Exploration in Cognition. San Francisco: W.H. Freeman, 1975.

Norman D.A., & Shallice, T. Attention to Action: Willed and Automatic Control of Behavior. Technical Report, Office of Naval Research, N00014-79-C-0323, 1980.

Robinson, A.L. More people are talking to computers as speech recognition enters the real world. Science, 1979, 203, pp. 634-638.

Robinson, A.L. Communicating with computers by voice. Science, 1979, 203, pp. 734-736.

Rumelhart, D.E., & Norman, D.A. Accretion Tuning and Restructuring: Three Modes of Learning. Technical Report 7602 University of California, San Diego, August 1976.

Schank, R., & Abelson, R. Scripts, Plans, Goals, and Understanding. Hillsdale, New Jersey: L. Erlbaum & Assoc., 1975

Schneider, W., & Shiffrin, R.M. Controlled and automatic human processing: I. Detection, search, and attention. Psychological Review, 1977, 84, pp. 1-66.

Siklossy, L. Let's Talk LISP. Englewood Cliffs, N.J.: Prentice-Hall, 1976.

Simmons, R.F. Semantic Networks: Their computation and use for understanding English sentences. In R.C. Schank, & K.M. Colby (Eds.), Computer Models of Thought and Language, San Francisco: W.H. Freeman & Co., 1973.

Simon H.A. Cognitive Science: The newest science of the artificial. In D.A. Norman (Ed.), Perspectives on Cognitive Science, Norwood, New Jersey, Ablex Publishing Corp., 1981.

Smith, E.E., Shoben, E.J., & Rips, L.J. Structure and process in semantic memory: A featural model for semantic decision. Psychological Review, 1974, 81, pp. 214-241.

Sternberg, S. Memory scanning: Mental processes revealed by reaction time experiments. American Scientist, 1969, 57, pp. 421-457.

Sternberg, S. High speed scanning in human memory. Science, 1966, 153, pp. 652-654.

Treisman, A.M. Contextual cues in selective listening. Quarterly Journal of Experimental Psychology, 1960, 12, pp. 242-248.

Tulving, E. Episodic and semantic memory. In E. Tulving, & W. Donaldson (Eds.). Organization of Memory. New York: Academic Press, 1972.

Waltz, D. Understanding line drawings of scenes with shadows. In P.H. Winston (Ed.), The Psychology of Computer Vision, New York: McGraw-Hill, 1975.

Weissman, C. LISP 1.5 Primer. Belmont, California: Dickenson Publishing Co., 1967.

Wescourt, K.T., Beard, M., Gould, L., & Barr, A. Knowledge-based CAI: CINS for Individualized Curriculum Sequencing. Technical Report #290, Institute for Mathematical Studies in the Social Sciences, Stanford University, October 1977.

Winston, P.H. Artificial Intelligence. Addison-Wesley Publishing Co., 1979.

Winston, P.H., & Horn, B.K. LISP. Reading, Mass.: Addison-Wesley Publishing Co., 1981.

Woods, W. Transition network grammars for natural language analysis. Communications of the ACM, 1970, 13, pp. 591-606.

APPENDIX A

TECHNICAL OBJECTIVES AND ACCOMPLISHMENTS

This current project has been targeted at examining the possibility for enhancing the pedagogical potential of voice-based training systems through the application of Artificial Intelligence (AI) theory and technology, as well as current developments in psychological theory.

Through the implementation of psychological cognitive theory via basic AI techniques, we have developed a simulated student whose behavior in a representative simulated task environment emulates essential features of real students. Again through the application of basic AI techniques we also developed a computer resident automated instructor which showed an ability to determine possible covert cognitive causes of overt student behavior, and to act on this knowledge to design individualized curricula. It is felt that this investigation yielded evidence supporting the logical sufficiency of the use of AI techniques for enhancing the pedagogical potential of voice-based training systems.

We initially proposed to produce:

1. A documented set of routines and techniques.

2. Data summaries on the performance of the routines.

3. References regarding the sources of our ideas.

4. Documentation of the relevant terminology.

5. A systematic explanantion for our rational in development.

6. Implications and sources of ideas for further development.

7. A set of research plans as to how the algorithms and techniques might be implemented and tested in actual trainers.

We suggested at that time, the we would be "spreading ourselves thin in attempting to develop several components in a short time span." Nevertheless, we believe we accomplished all of the production goals listed. There are specific section of this report relating to the first six stated objectives. Information concerning the last objective can generally be found throughout the report.

INDEX

DISTRIBUTION LIST

Commanding Officer
Naval Training Equipment Center
Orlando, FL 32813 50

Defense Technical Information Center
Cameron Station
Alexandria, VA 22310 12

(All others receive one copy)

Commander
USAISD
Attn: Library
Fort Devens, MA 01433

National Aviation Facilities
Experimental Center
Library
Atlantic City, NJ 08405

Dr. Donald W. Connolly
Research Psychologist
Federal Aviation Administration
NAFEC ANA-230
Atlantic City, NJ 08405

Mr. Leahmond Tyre
Fleet Material Support Office
Code 9333
Mechanicsburg, PA 17055

T. Weiner
Naval Air Development Center
Code 4043
Warminster, PA 18974

Dr. Christian Skriver
Naval Air Development Center
Code 6021
Warminster, PA 18974

Dr. Julie A. Hopson
Naval Air Development Center
Code 6021
Warminster, PA 18974

Chief, Research Office
Office Deputy Chief of Staff for
Personnel
Department of Army
Washington, DC 20310

HQAFSC/OLS
Andrews AFB, DC 20334

Chief of Naval Operations
OP-98711
Attn: Dr. R. G. Smith
Washington, DC 20350

Chief of Naval Operations
OP-115
Research Development & Studies
Room 0836
Washington, DC 20350

Chief of Naval Operations
OP-39T
Washington, DC 20350

Chief of Naval Operations (OP-596)
Washington, DC 20350

Commander
Naval Air Systems Center
Air 413F
Washington, DC 20361

Scientific Advisor
Headquarters US Marine Corps
Washington, DC 20380

Commander
Naval Air Test Center
CT 176
Patuxent River, MD 20670

Dr. Sam Schiflett
Naval Air Test Center
SY 721
Patuxent River, MD 20670

Director Educational Development
Academic Computing Center
US Naval Academy
Annapolis, MD 21402

Dr. Tice De Young
US Army Engineer Topographic Lab
Research Institute
Ft. Belvoir, VA 22060

Dr. Jesse Orlansky
Institute for Defense Analyses
Science and Technology Division
400 Army Navy Drive
Arlington, VA 22202

Chief of Naval Research
Code 455
800 N. Quincy St.
Arlington, VA 22217

Chief of Naval Research
Code 458
800 N. Quincy St.
Arlington, VA 22217

Dr. Marshall J. Farr
Director, Personnel and Training
Research Program (Code 458)
Office of Naval Research
Arlington, VA 22217

Dr. Henry M. Halff
Office of Naval Research
Code 442
Arlington, VA 22217

Dr. Henry J. Dehaan
US Army Research Institute
5001 Eisenhower AV
Alexandria, VA 22333

Commander
Naval Air Force
US Atlantic Fleet (Code 311)
NAS Norfolk
Norfolk, VA 23511

Army Training Support Center
ATTSC-OS
St. Eustis, VA 23604

US Army Signal Center
Fort Gordon, GA 30905

Chief of Naval Education and Training
Code N-9
Pensacola, FL 32508

Chief
ARI Field Unit
P. O. Box 476
Ft. Rucker, AL 36362

LSO Training Model Manager
Officer in Charge
LSO School
Box 171
NAS Cecil Field, FL 32215

Commanding Officer
Naval Air Technical Training Center
Code 104, Building S-54
NAS Memphis (85)
Millington, TN 38054

Mr. Harold A. Kottmann
ASD/AXA
Attn: W. R. Vivians
Wright-Patterson AFB. OH 45433

5570 AMRL/HEC
Wright-Patterson AFB, 45433

Mr. Don F. McKechnie
Research Psyshologist
AFAMRL/HEF
Wright-Patterson AFB
Dayton, OH 45433

Air Force Human Resources Laboratory
AFHRL/LR
Logistics Research Division
Wright-Patterson AFB, OH 45433

ASO/ENETC
Mr. R. G. Cameron
Wright-Patterson AFB, OH 45433

Commanding Officer
Naval Hospital Corps School
Great Lakes, IL 60088

Headquarters
Air Training Command,
TSZ
Brooks AFB, TX 78235

US Air Force Human Resources Lab
AFHRL-OT (Dr. Rockway)
Williams AFB, AZ 85224

Chief of Naval Education and Training
Liaison Office
Human Resource Laboratory
Flying Training Division
Williams AFB, AZ 85224

US Air Force Human Resources Lab
AFHRL-OT
Operational Training Division
Williams AFB, AZ 85224

Commanding Officer
Human Resources Laboratory
Operational Training Division
Williams AFB, AZ 85224

US Air Force Human Resources Lab
AFHRL-FT
Flying Training Division
Attn: Dr. Edwards
Williams AFB, AZ 85224

Commander
Naval Air Force
US Pacific Fleet (Code 316)
NAS North Island
San Diego, CA 92135

Commanding Officer
Fleet Training Center
ATTN: Training Department
Naval Station
San Diego, CA 92136

Commanding Officer
Fleet Combat Training Center, Pacific
Code 33A
San Diego, CA 92147

Commandant
US Army Field Artillery School
Counterfire Department
Attn: Mr. Willie Martinez
FT. Sill, OK 73503

Mr. Melvin C. Moy
Navy Personnel Research & Develop. Ctr
Information and Decision Processing
Code 305
San Diego, CA 92152

Navy Personnel Research and Development
Center
Attn: M. MC Dowell
Library, Code P201L
San Diego, CA 92152

John Silva
Naval Ocean Systems Center
Code 823
San Diego, CA 92152

Dr. Robert A Wisner
Navy Personnel Research and Development Center
Code P309
San Diego, CA 92152

Commander
Naval Weapons Center
Code 3154
Attn: Mr. Bob Curtis
China Lake, CA 93555

Director
US Army Research & Technology Lab
Davol-AS (Attn: Dr. R. S. Dunno)
207-S Ames Research Center
Moffett Field, CA 94035

Mr. Gary Poock
Naval PG School
Code 55PK
Monterey, CA 93940